

NASA Technical Memorandum 103240  
ICOMP-90-20

# Efficient Implementation of Minimal Polynomial and Reduced Rank Extrapolation Methods

Avram Sidi  
*Technion—Israel Institute of Technology*  
*Haifa, Israel*

*and Institute for Computational Mechanics in Propulsion*  
*Lewis Research Center*  
*Cleveland, Ohio*

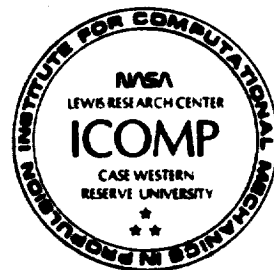
(NASA-TM-103240) EFFICIENT IMPLEMENTATION  
OF MINIMAL POLYNOMIAL AND REDUCED RANK  
EXTRAPOLATION METHODS (NASA) 43 p CSCL 12A

NPO-26610

Unclas  
63/84 0297011

August 1990

**NASA**





# **EFFICIENT IMPLEMENTATION OF MINIMAL POLYNOMIAL AND REDUCED RANK EXTRAPOLATION METHODS<sup>†</sup>**

**Avram Sidi  
Computer Science Department  
Technion - Israel Institute of Technology  
Haifa 32000, Israel**

**and Institute for Computational Mechanics in Propulsion  
Lewis Research Center  
Cleveland, Ohio 44135**

## **A B S T R A C T**

The minimal polynomial extrapolation (MPE) and reduced rank extrapolation (RRE) are two very effective techniques that have been used in accelerating the convergence of vector sequences, such as those that are obtained from iterative solution of linear and nonlinear systems of equations. Their definitions involve some linear least squares problems, and this causes difficulties in their numerical implementation. In this work timewise efficient and numerically stable implementations for MPE and RRE are developed. A computer program written in FORTRAN 77 is also appended and applied to some model problems.

E-5658

---

<sup>†</sup>Work funded under Space Act Agreement C99066G.

## 1. INTRODUCTION

The minimal polynomial extrapolation (MPE) of Cabay and Jackson [2] and the reduced rank extrapolation (RRE) of Eddy [3] and Mešina [9] are two methods used in accelerating the convergence of a large class of vector sequences. In particular, they are employed for accelerating the convergence of fixed point iterative techniques for linear or nonlinear systems of equations, such as those that arise in the discrete solution of continuum problems.

A unified treatment of these and other extrapolation methods has been given in the survey paper of Smith, Ford, and Sidi [19], where some numerical testing for them is also provided. Detailed convergence analyses for MPE and RRE have been presented in Sidi [12], Sidi and Bridger [16], and Sidi [13], and we shall mention some of the results that follow from these analyses later in this work. Also, both MPE and RRE are very closely related to some well known Krylov subspace methods when they are applied to linearly generated vector sequences, and this subject is explored in detail in [13]. In fact, MPE and RRE are equivalent to the Arnoldi method and generalized conjugate residuals (GCR), respectively, when they are all applied to linear systems of equations starting with the same initial approximation. For the method of Arnoldi see Saad [10], and for GCR see Eisenstat, Elman, and Schultz [4]. We also mention that the conjugate gradient type method of Axelsson [1], the method of Young and Jea [22] that has been called ORTHODIR, and the recent generalized minimal residual method (GMRES) of Saad and Schultz [11] are all equivalent to GCR, and are used in solving linear equations. Recursion relations that exist amongst various approximations that are obtained from both methods are discussed in the paper by Ford and Sidi [6], where the existence of an interesting four-term lozenge recursion is shown. MPE and RRE have been employed successfully in Sidi and Celestina [17] in accelerating the convergence of some finite difference solution techniques in large scale computational fluid dynamics problems. Finally, the application of MPE and RRE and other vector extrapolation methods to the iterative solution of consistent singular linear systems has been considered in Sidi [15], where this approach is shown to be sound theoretically, and precise convergence analyses are also provided.

The definitions of MPE and RRE involve the solution of a linear least squares problem, the number of equations in this problem being equal to the dimension of the vectors in the given sequence. Since, in general, this dimension may be very large, as it is, for example, in three-dimensional computational fluid dynamics problems, the matrix of the least squares problem may be

very large. Thus, if standard linear least squares packages are used, the time and core memory requirements in the implementation of MPE and RRE may become prohibitive. To circumvent this problem, the solution of the linear least squares problem was achieved in [17] by solving the corresponding normal equations that is much less costly than using least squares packages. This approach proves to be quite efficient when the amount of extrapolation is not very large. When the amount of extrapolation is increased, however, the accuracy decreases, as the normal equations become very ill conditioned.

In the present work we propose new implementations for MPE and RRE, which are very inexpensive as far as both time and core memory requirements are concerned, and are stable numerically as the amount of extrapolation is increased. These implementations are also quite interesting mathematically, as they allow one to compute exactly (or estimate) the accuracy achieved in the extrapolation process without actually computing the residuals at each stage. This can be employed to further reduce the cost of implementation.

The plan of this paper is as follows: In Section 2 we briefly review the definitions of MPE and RRE. In Section 3 we consider the application of MPE and RRE to vector sequences that are generated by iterative solution of linear systems as this provides the motivation for different modes of usage of the methods. We devote Sections 4-6 to the development of the new implementations of MPE and RRE and the description of the mathematical features of these implementations. In Section 4 we give the details of the new implementations. One of the crucial ingredients of these implementations is the efficient solution of the least squares problems by use of QR factorization. In Section 5 we show how, in these new implementations, the  $l_2$ -norms of the residuals can be computed exactly for linear systems (or estimated for nonlinear systems) without doing extra vector computations. This enables us to assess the accuracy of the extrapolation without actually carrying it out, and can be used to reduce the amount of computation drastically. In Section 6 we discuss the operation counts and the storage requirements for the new implementations. In Section 7 we discuss some practical matters concerning the efficient use of MPE or RRE or any other vector extrapolation methods. Finally, in Section 8 we give some numerical results obtained by applying MPE and RRE through their new implementations to certain model problems. A computer program written in FORTRAN 77 that implements MPE and RRE is provided in the appendix.

## 2. REVIEW OF MPE AND RRE

Let  $x_0, x_1, x_2, \dots$  be a given sequence of  $N$ -dimensional column vectors, and denote its limit or antilimit by  $s$ . The vectors  $x_j$  are assumed to be complex, in general. Define

$$u_i = \Delta x_i = x_{i+1} - x_i \quad \text{and} \quad w_i = \Delta u_i = \Delta^2 x_i, \quad i = 0, 1, 2, \dots \quad (2.1)$$

Define the  $N \times (j+1)$  matrices  $U_j^{(n)}$  and  $W_j^{(n)}$  by

$$U_j^{(n)} = [u_n \mid u_{n+1} \mid \dots \mid u_{n+j}] \quad (2.2)$$

and

$$W_j^{(n)} = [w_n \mid w_{n+1} \mid \dots \mid w_{n+j}] \quad (2.3)$$

### 2.1 Definition of MPE

For MPE the approximation  $s_{n,k}$  to  $s$ , the desired limit or antilimit, is defined by

$$s_{n,k} = \sum_{j=0}^k \gamma_j x_{n+j} \quad (2.4)$$

where the  $\gamma_j$  are determined as follows:

- (i) Use the least squares method to solve the overdetermined and, in general, inconsistent linear system

$$U_{k-1}^{(n)} c = -u_{n+k} \quad (2.5)$$

where  $c = (c_0, c_1, \dots, c_{k-1})^T$ .

- (ii) Set  $c_k = 1$ , and compute the  $\gamma_j$  by

$$\gamma_j = \frac{c_j}{\sum_{i=0}^k c_i}, \quad 0 \leq j \leq k, \quad (2.6)$$

assuming that  $\sum_{i=0}^k c_i \neq 0$ . When this condition is not satisfied  $s_{n,k}$  does not exist.

### 2.2 Definition of RRE

For RRE the approximation  $s_{n,k}$  to  $s$ , the desired limit or antilimit, is defined by

$$s_{n,k} = x_n + \sum_{i=0}^{k-1} \xi_i u_{n+i} \quad (2.7)$$

where the  $\xi_i$  are determined by solving the overdetermined and, in general, inconsistent linear system

$$W_{k-1}^{(n)} \xi = -u_n , \quad (2.8)$$

with  $\xi = (\xi_0, \xi_1, \dots, \xi_{k-1})^T$ , using the least squares method. Since a least squares solution to (2.8) always exists,  $s_{n,k}$  always exists. In particular,  $s_{n,k}$  exists uniquely when the matrix  $W_{k-1}^{(n)}$  has full rank, i.e.,  $\text{rank}(W_{k-1}^{(n)}) = k$ , or, equivalently, when the vectors  $w_n, w_{n+1}, \dots, w_{n+k-1}$  are linearly independent. It can easily be shown that  $\text{rank}(W_{k-1}^{(n)}) = k$ , thus  $s_{n,k}$  exists uniquely, when  $\text{rank}(U_k^{(n)}) = k+1$ .

There exists an equivalent formulation of RRE that seems to be more suitable for computer implementation. It also has the advantage of unifying most of the algorithmic aspects of MPE and RRE. In this formulation  $s_{n,k}$  is of the form given in (2.4); only this time the  $\gamma_j$  are obtained by the least squares solution of the overdetermined and, in general, inconsistent linear system

$$U_k^{(n)} \gamma = 0 , \quad (2.9)$$

where  $\gamma = (\gamma_0, \gamma_1, \dots, \gamma_k)^T$ , subject to the constraint

$$\sum_{j=0}^k \gamma_j = 1 . \quad (2.10)$$

(Note that the  $\gamma_j$  in MPE satisfy (2.10) automatically, as can easily be seen from (2.6).)

#### Remarks:

- (1) It is important to realize that the  $\gamma_j$  in  $s_{n,k} = \sum_{j=0}^k \gamma_j x_{n+j}$  depend on both  $n$  and  $k$ .
- (2) In most applications,  $N$ , the dimension of the vectors  $x_i$ , is much larger than  $k$ , so that the matrices  $U_j^{(n)}$  have many more rows than columns. Therefore, there is great need to reduce the amount of numerical work with the columns of the matrices  $U_j^{(n)}$ .

### 3. APPLICATION OF MPE AND RRE TO LINEAR SYSTEMS

Consider the linear nonsingular  $N$ -dimensional linear system

$$x = Ax + b , \quad (3.1)$$

where  $A$  is an  $N \times N$  matrix and  $b$  is an  $N$ -dimensional column vector. Pick an initial vector  $x_0$ , and generate the vectors  $x_1, x_2, \dots$ , by the iterative scheme

$$x_{i+1} = Ax_i + b, \quad i = 0, 1, \dots \quad (3.2)$$

The solution  $s$  of (3.1) is now the limit of the sequence  $x_0, x_1, x_2, \dots$ , when the latter converges, otherwise,  $s$  is the antilimit.

Let  $k_0$  be the degree of the minimal polynomial of the matrix  $A$  with respect to the vector  $x_n - s$ . Then the following statements are true.

- (i)  $s_{n,k_0}$  is uniquely defined both for MPE and RRE, and

$$s_{n,k_0} = s. \quad (3.3)$$

Also the linear systems in (2.5), (2.8), and (2.9) are consistent for  $k = k_0$ , even though they may be overdetermined. This is a consequence of the fact that the vectors  $u_{n+j}$ ,  $0 \leq j \leq k_0 - 1$ , are linearly independent, and  $u_{n+k_0}$  lies in their span. (See [13, Section 2.2].)

- (ii) For  $k < k_0$ ,  $s_{n,k}$  is uniquely defined for RRE. For MPE, however,  $s_{n,k}$  may fail to exist when  $k < k_0$ . When the matrix  $C = I - A$  has positive definite hermitian part,  $s_{n,k}$  exists uniquely for MPE also for  $k < k_0$ . (See [13, Section 2.2].) More generally,  $s_{n,k}$  exists uniquely for MPE also for  $k < k_0$ , if the eigenvalues of  $C$  all lie on one side of a straight line through the origin in the complex plane, or, equivalently, if they all lie in an open sector  $S = \{\mu: |\arg \mu - \theta| < \pi/2\}$ , for some  $\theta$ ,  $-\pi < \theta \leq \pi$ . This result can be proved exactly as Theorem 2.2 in [13] with  $C$  there replaced by  $e^{-i\theta}C$ .
- (iii) When the Arnoldi method and GCR are used in solving the linear system  $Cx = b$ , where  $C = I - A$ , with  $x_n$  as the initial vector, they become equivalent to MPE and RRE, respectively. Specifically, the approximations obtained from the Arnoldi method and GCR are exactly  $s_{n,1}, s_{n,2}, \dots$ , that are produced by MPE and RRE, respectively. (See [13, Section 2.3].)
- (iv) If the *distinct nonzero* eigenvalues of  $A$  are denoted  $\lambda_j$ ,  $j = 1, 2, \dots$ , and are ordered such that

$$|\lambda_1| \geq |\lambda_2| \geq |\lambda_3| \geq \dots, \quad (3.4)$$

then, provided

$$|\lambda_k| > |\lambda_{k+1}|, \quad (3.5)$$

and  $A$  is diagonalizable, we have

$$s_{n,k} - s = O(|\lambda_{k+1}|^n) \quad \text{as } n \rightarrow \infty, \quad (3.6)$$

both for MPE and RRE. (The coefficient of  $|\lambda_{k+1}|^n$  on the right hand side of (3.6) becomes large when the largest eigenvalues  $\lambda_1, \lambda_2, \dots$ , are close to 1.) In view of the fact that



$x_n - s = O(|\lambda_1|^n)$  as  $n \rightarrow \infty$ , we conclude that MPE and RRE are both true acceleration methods. Under the same conditions, if  $s_{n,k} - s$  is precisely  $O(|\lambda_{k+1}|^n)$  as  $n \rightarrow \infty$ , then, the  $\gamma_j$  for MPE and RRE are such that

$$P^{n,k}(\lambda) = \sum_{j=0}^k \gamma_j \lambda^j = \prod_{i=1}^k \frac{\lambda - \lambda_i}{1 - \lambda_i} + O(|\lambda_{k+1}/\lambda_k|^n) \quad \text{as } n \rightarrow \infty, \quad (3.7)$$

i.e., for fixed  $k$  and for all sufficiently large  $n$ , the polynomial  $P^{(n,k)}(\lambda)$  has precisely  $k$  zeros that tend to  $\lambda_1, \lambda_2, \dots, \lambda_k$ . Furthermore, if we denote the zero of  $P^{(n,k)}(\lambda)$  that tends to  $\lambda_j$  by  $\lambda_j(n)$ , then

$$\lambda_j(n) - \lambda_j = O(|\lambda_{k+1}/\lambda_j|^n) \quad \text{as } n \rightarrow \infty, \quad 1 \leq j \leq k. \quad (3.8)$$

The proofs of (3.6) and (3.7) have been given in [12, Sections 3 and 4]. The proof of (3.8) will be published in the future. In case the matrix  $A$  in (3.2) is normal, the right hand sides of (3.7) and (3.8) can be replaced by  $O(|\lambda_{k+1}/\lambda_k|^{2n})$  and  $O(|\lambda_{k+1}/\lambda_j|^{2n})$ , respectively. (The result in (3.6) remains the same, however.) This implies that when  $A$  is normal the rates of converge of  $P^{(n,k)}(\lambda)$  and its zeros  $\lambda_j(n)$  are twice those that can be achieved otherwise. These results follow from the corresponding results of Sidi [14].

For the most general case in which the matrix  $A$  is not diagonalizable, the results in (3.6)-(3.8) need to be modified considerably. For a complete treatment of this case see [16, Sections 2,3, and 5], where modifications of (3.6) and (3.7) are given. The modification of (3.8) will be published in the future.

A direct consequence of the result given in (3.6) is that better accuracy may be obtained if extrapolation is preceded by a number of fixed point iterations. This has indeed been observed numerically both for linear and nonlinear problems. We shall comment on this again in Section 7.

- (v) Let us denote  $C = I - A$ . Then  $s$  is the solution to  $Cx = b$ . Denote by  $\pi_k$  the set of all polynomials  $Q_k(\lambda)$  of degree at most  $k$  that satisfy  $Q_k(0) = 1$ . Consider now  $s_{n,k}$  as obtained by applying MPE or RRE to the vector sequence  $x_0, x_1, \dots$ . Then

$$\|r(s_{n,k})\| \leq \left( \min_{Q_k \in \pi_k} \|Q_k(C)\| \right) \|r(x_n)\| \quad \text{for RRE,} \quad (3.9)$$

where  $r(x) = Ax + b - x = b - Cx = -C(x - s)$  is the residual for  $x$ , and  $\|\cdot\|$  is the  $l_2$  vector norm,

or the matrix norm induced by it. (In fact,  $\|r(s_{n,k})\|$ ,  $k = 0, 1, 2, \dots$ , is a monotonically decreasing sequence for RRE.) Similarly, if  $C_h = \frac{1}{2}(C + C^*)$ , the hermitian part of  $C$ , is positive definite, then

$$\|C_h^{1/2}(s_{n,k} - s)\| \leq \beta \left( \min_{Q_k \in \pi_k} \|Q_k(C)\| \right) \|C_h^{1/2}(x_n - s)\| \quad \text{for MPE,} \quad (3.10)$$

where

$$\beta = \begin{cases} L & \text{if } C \text{ is normal} \\ L\sqrt{\text{cond}(C_h)} & \text{otherwise,} \end{cases} \quad (3.11)$$

with  $L = \|C_h^{-1/2} C C_h^{-1/2}\| \geq 1$ . Note that both  $\|r(x)\|$  and  $\|C_h^{1/2}x\|$  are true norms for  $x$ . Two types of bounds for  $\min_{Q_k \in \pi_k} \|Q_k(C)\|$ , in case  $C_h$  is positive definite, are given in [13, Section 4], and these can be used to derive upper bounds for  $\|r(s_{n,k})\|$  and  $\|C_h^{1/2}(s_{n,k} - s)\|$  for fixed  $n$  and increasing  $k$ . For details see [13]. These bounds are employed in [17] to justify the use of the extrapolation strategy that has been called "cycling" in [19] and all subsequent publications.

Finally, analogous and almost identical results exist for the case in which the system in (3.1) is singular but consistent, so that it has an infinity of solutions. In this case the limit or antilimit depends on  $x_0$  in a very specific manner. For details, see [15].

**Remark:** The various Krylov subspace methods like the Arnoldi method and GCR and others can be applied only to linear systems. Acceleration methods such as MPE and RRE, however, can be applied to nonlinear systems as well as linear ones. The reason for this is that, unlike the Krylov subspace methods, MPE and RRE are defined exclusively in terms of the given vector sequence, which may be generated, for example, by an iterative method. Whether the vector sequence is generated linearly or nonlinearly is irrelevant to the definitions of MPE and RRE and other vector extrapolation methods. This is a very important property of vector extrapolation methods.

## 4. IMPLEMENTATION OF MPE AND RRE

### 4.1 General Considerations

As we have seen in Section 2, both MPE and RRE entail linear least squares problems in their definitions. There is, therefore, an immediate need for the efficient solution of these problems. We

propose to solve these problems by applying the QR factorization to the matrices  $U_k^{(n)}$ .

To keep the notation simple we shall set  $n = 0$  everywhere, and denote the matrices  $U_j^{(0)}$  by  $U_j$ . This amounts to simply renaming  $x_n$  and calling it  $x_0$ .

We assume that the vectors  $u_0, u_1, \dots, u_k$  are linearly independent so that the  $N \times (k+1)$  matrix  $U_k$  is of full rank  $k+1$ . The case in which  $u_0, u_1, \dots, u_k$  are linearly dependent will be discussed later in this section. We recall that for the linear system in (3.1) this assumption is valid when  $k < k_0$ , where  $k_0$  is the degree of the minimal polynomial of the matrix  $A$  with respect to the vector  $x_0$ -s. Therefore, there is a unique  $N \times (k+1)$  matrix  $Q_k$ ,

$$Q_k = [q_0 | q_1 | \dots | q_k], \quad (4.1)$$

whose columns  $q_i$  satisfy

$$(q_i, q_j) = q_i^* q_j = \delta_{ij}, \quad (4.2)$$

and a unique  $(k+1) \times (k+1)$  upper triangular matrix  $R_k$ ,

$$R_k = \begin{bmatrix} r_{00} & r_{01} & r_{02} & \dots & r_{0k} \\ & r_{11} & r_{12} & \dots & r_{1k} \\ & & r_{22} & \dots & r_{2k} \\ & 0 & & \ddots & \\ & & & & r_{kk} \end{bmatrix}, \quad (4.3)$$

with  $r_{ii} > 0$ ,  $i = 0, 1, \dots, k$ , such that

$$U_k = Q_k R_k. \quad (4.4)$$

This QR factorization amounts to orthonormalizing the vectors  $u_0, u_1, u_2, \dots$ , in this order. It is important to retain this order, as this enables us to form the QR factorization of  $U_{k+1}$  by appending one additional column to  $Q_k$  to obtain  $Q_{k+1}$ , and a corresponding column to  $R_k$  to obtain  $R_{k+1}$ . Needless to say, this results in considerable savings in computing time.

QR factorization can be performed in different ways. The simplest way is the Gram-Schmidt (GS) process for orthonormalization of  $u_0, u_1, u_2, \dots$ . This process is very unstable, however, in the sense that the computed vectors  $q_0, q_1, q_2, \dots$ , are very far from being orthogonal. The modified Gram-Schmidt (MGS) process, on the other hand, seems to be quite stable, and is the one that we have

preferred. We recall that MGS is entirely equivalent to GS mathematically, and requires the same number of arithmetic operations as GS. The two methods are different numerically, however. For details, see, e.g., Golub and Van Loan [7, pp. 218-219].

For the sake of completeness we describe MGS for the case in which the vectors  $u_0, u_1, u_2, \dots$ , are introduced one by one and in this order.

### Algorithm MGS

Step 1. read  $u_0$ , and compute the scalar  $r_{00}$  and the vector  $q_0$  according to  
 $r_{00} = (u_0, u_0)^{1/2}$  and  $q_0 = u_0 / r_{00}$ .

Step 2. for  $k = 1, 2, \dots$ , do  
 read  $u_k$ , and set  $u_k^{(0)} = u_k$   
 for  $j=0$  to  $k-1$  do  
 $r_{jk} = (q_j, u_k^{(j)})$   
 $u_k^{(j+1)} = u_k^{(j)} - r_{jk} q_j$   
 end  
 compute  $r_{kk}$  and  $q_k$  according to  
 $r_{kk} = (u_k^{(k)}, u_k^{(k)})^{1/2}$  and  $q_k = u_k^{(k)} / r_{kk}$   
 end

(Here  $(y, z)$  stands for the Euclidean inner product  $y^* z$ , as before.)

It is easy to see that, when implementing MGS on a computer,  $u_k^{(0)}, u_k^{(1)}, \dots, u_k^{(k)}$ , and  $q_k$  can all be made to occupy the same storage locations. As we shall see in the next paragraph, the computation of  $s_{0,k}$  can be based on the  $q_j$  without the need to save either the  $x_j$  or the  $u_j$ . We can thus let  $u_k$  occupy the same storage locations as the  $u_k^{(i)}$ .

QR factorization can also be achieved by using Householder transformations. Although the computed matrices  $Q_k$  produced in this approach are closer to unitary than those produced by MGS when the  $l_2$  condition number of  $U_k$  is large, the amount of computing in this approach is about twice that required by MGS. We shall elaborate on this further in Section 7.

We now recall from the definitions of MPE and RRE, that the approximations  $s_{0,k}$  for both methods can be expressed in the form

$$s_{0,k} = \sum_{j=0}^k \gamma_j x_j \quad \text{with} \quad \sum_{j=0}^k \gamma_j = 1. \quad (4.5)$$

Assuming that  $\gamma_0, \gamma_1, \dots, \gamma_k$  have been determined, let us compute  $\xi_0, \xi_1, \dots, \xi_{k-1}$  from

$$\xi_0 = 1 - \gamma_0 \quad \text{and} \quad \xi_j = \xi_{j-1} - \gamma_j, \quad 1 \leq j \leq k-1. \quad (4.6)$$

Then, we can reexpress  $s_{0,k}$  in the form

$$s_{0,k} = x_0 + \sum_{i=0}^{k-1} \xi_i u_i = x_0 + U_{k-1} \xi, \quad (4.7)$$

where  $\xi = (\xi_0, \xi_1, \dots, \xi_{k-1})^T$ . Substituting now  $U_{k-1} = Q_{k-1} R_{k-1}$  in (4.7), we obtain

$$s_{0,k} = x_0 + Q_{k-1} (R_{k-1} \xi) = x_0 + \sum_{j=0}^{k-1} \eta_j q_j, \quad (4.8)$$

where

$$\eta_j = (j+1)\text{st component of the column vector } R_{k-1} \xi, \quad j = 0, 1, \dots, k-1. \quad (4.9)$$

This approach to the computation of  $s_{0,k}$  is very advantageous, as it enables us to overwrite  $x_1, x_2, \dots$ , and  $u_0, u_1, \dots$ , and thus saves a lot of storage.

#### 4.2 Determination of the $\gamma_j$ When rank $(U_k) = k+1$

The only thing that remains to be done now is to determine the  $\gamma_j$ , and this requires separate treatments for MPE and RRE.

##### 4.2.1 Determination of the $\gamma_j$ for MPE

As mentioned in Section 2, in order to determine the  $\gamma_j$  for  $s_{0,k}$  in MPE we first solve the over-determined system

$$U_{k-1} c = -u_k \quad (4.10)$$

by least squares. Since we also assume that the rank of  $U_k$  is  $k+1$ , we conclude that  $c$  is the unique solution of the normal equations

$$U_{k-1}^* U_{k-1} c = -U_{k-1}^* u_k. \quad (4.11)$$

Upon invoking  $U_{k-1} = Q_{k-1} R_{k-1}$  in (4.11) and using the fact that  $Q_{k-1}^* Q_{k-1} = I_{k \times k}$  = the  $k \times k$  identity matrix, and the fact that  $R_{k-1}$  is a nonsingular matrix, we obtain

$$R_{k-1} c = -Q_{k-1}^* u_k. \quad (4.12)$$

It is easy to see that

$$Q_{k-1}^* u_k = (r_{0k}, r_{1k}, \dots, r_{k-1,k})^T \equiv \rho_k, \quad (4.13)$$

so that (4.12) becomes

$$R_{k-1}c = -\rho_k. \quad (4.14)$$

This is a linear system of  $k$  equations in the  $k$  unknowns  $c_0, c_1, \dots, c_{k-1}$ , and its matrix  $R_{k-1}$  is upper triangular. Hence its solution can be achieved easily by back substitution.

Once  $c_0, c_1, \dots, c_{k-1}$  are determined, we set  $c_k = 1$ , and compute the  $\gamma_j$  from (2.6), provided

$$\sum_{i=0}^k c_i \neq 0.$$

#### 4.2.2 Determination of the $\gamma_j$ for RRE

Again as we mentioned in Section 2, the  $\gamma_j$  for  $s_{0,k}$  in RRE can be determined by solving the overdetermined system

$$U_k \gamma = 0 \quad (4.15)$$

by least squares subject to the constraint

$$\sum_{j=0}^k \gamma_j = 1. \quad (4.16)$$

This amounts to minimizing the positive definite quadratic form  $\gamma^* U_k^* U_k \gamma$  subject to (4.16). Consequently, the lemma in Appendix A applies, and the  $\gamma_j$  can be obtained by solving the linear system of  $k+2$  equations

$$\begin{aligned} U_k^* U_k \gamma &= \lambda \tilde{e} \\ \sum_{j=0}^k \gamma_j &= 1 \end{aligned} \quad (4.17)$$

for  $\gamma_0, \gamma_1, \dots, \gamma_k$ , and  $\lambda$ . Here

$$\tilde{e} = (1, 1, \dots, 1)^T. \quad (4.18)$$

As is stated in the same lemma,  $\lambda$  turns out to be strictly positive, and is given by

$$\lambda = \gamma^* U_k^* U_k \gamma \quad \text{at the solution.} \quad (4.19)$$

The  $\gamma_j$  can be obtained by first solving the linear system

$$U_k^* U_k d = \tilde{e} \quad (4.20)$$

for  $d = (d_0, d_1, \dots, d_k)^T$ , and letting

$$\lambda = \left[ \sum_{j=0}^k d_j \right]^{-1} . \quad (4.21)$$

and finally setting

$$\gamma = \lambda d . \quad (4.22)$$

As far as the solution of the system in (4.20) is concerned, we accomplish this again by using the QR factorization of  $U_k$ . Again by  $Q_k^* Q_k = I_{(k+1) \times (k+1)}$  = the  $(k+1) \times (k+1)$  identity matrix, we can rewrite (4.20) in the form

$$R_k^* R_k d = \tilde{e} . \quad (4.23)$$

This system can be solved by forward and back substitution as the matrix  $R_k$  is upper triangular.

#### 4.3 Treatment of the Case $\text{rank}(U_k) = k$

Up to this point we discussed the case in which the vectors  $u_0, u_1, \dots, u_k$  are linearly independent. Since these vectors are being introduced one by one, we can view this case as adding the vector  $u_k$  to the linearly independent set  $\{u_0, u_1, \dots, u_{k-1}\}$  and obtaining the linearly independent set  $\{u_0, u_1, \dots, u_k\}$ . We now consider the case in which  $\{u_0, u_1, \dots, u_{k-1}\}$  is a linearly independent set, but  $\{u_0, u_1, \dots, u_k\}$  is not, i.e.,  $\text{rank}(U_k) = k$ . This exhibits itself through  $r_{kk} = 0$  in the QR factorization step.

If we are applying MPE, then we can compute the  $\gamma_j$  by solving the (nonsingular) system in (4.14) and employing (2.6), provided  $\sum_{i=0}^k c_i \neq 0$  there. We then compute  $s_{0,k}$ .

If we are applying RRE, we can compute  $s_{0,k}$  as follows: First, by the linear dependence of  $u_0, u_1, \dots, u_k$ , there exist constants  $\alpha_0, \alpha_1, \dots, \alpha_k$ , not all zero such that  $\sum_{i=0}^k \alpha_i u_i = 0$ . This implies that the linear system in (4.15) is consistent. Also we can write  $U_k = Q_k R_k$ , where  $Q_k$  and  $R_k$  are as in (4.1)-(4.3),  $q_0, q_1, \dots, q_{k-1}$  are uniquely determined and  $q_k$  is arbitrary in (4.1), and  $r_{kk} = 0$  in (4.3). Multiplying both sides of (4.15) by  $Q_k^*$ , and using the fact that  $Q_k^* Q_k = I_{(k+1) \times (k+1)}$ , we obtain the system of  $k+2$  equations

$$R_k \gamma = 0 \quad \text{and} \quad \sum_{j=0}^k \gamma_j = 1 . \quad (4.24)$$

Now, by  $r_{kk} = 0$ , this system actually consists of the  $k+1$  inhomogeneous equations

$$[R_{k-1} | \rho_k] \gamma = 0 \quad \text{and} \quad \sum_{j=0}^k \gamma_j = 1 \quad (4.25)$$

in the  $k+1$  unknowns  $\gamma_0, \gamma_1, \dots, \gamma_k$ . Since a least squares solution for the linear system in (2.8) always exists, a solution for the  $\gamma_j$  always exists too. Consequently, the equations in (4.25) always have a solution for RRE. Once we determine a set of  $\gamma_j$ 's, we compute  $s_{0,k}$ .

Comparing (4.25) with (4.14) and (2.6), we see that if  $s_{0,k}$  exists for MPE when  $\text{rank}(U_k) = k$ , then it is equal to  $s_{0,k}$  for RRE.

If the vector sequence  $x_0, x_1, x_2, \dots$ , is generated as in (3.2), then, as explained in Section 3,  $\text{rank}(U_k) = k+1$  for  $k < k_0$ , where  $k_0$  is the degree of the minimal polynomial of  $A$  with respect to  $x_0$ -s. The smallest value of  $k$  for which  $\text{rank}(U_k) = k$  is  $k_0$ , and at  $k = k_0$  we already reach the solution, i.e.,  $s_{0,k_0} = s$ . That is the first time  $r_{kk} = 0$  occurs, we have  $s_{0,k} = s$ , and stop.

If the vector sequence  $x_0, x_1, x_2, \dots$ , is not generated linearly, and  $\text{rank}(U_{k-1}) = k$ , but  $\text{rank}(U_k) = k < k+1$ , then we can compute  $s_{0,k}$  first, and then take  $s_{0,k}$  or a nearby vector as  $x_0$ , and restart the computation. Other strategies for continuing the computation can likewise be devised, but we shall not pursue this matter further.

It should be mentioned, however, that, due to roundoff, the chances of encountering the case  $\text{rank}(U_k) < k+1$  in practice are extremely small. We have thus not included the treatment of this case in the computer program given in the appendix.

#### 4.4 Summary of Implementations

We now summarize the major steps of the implementations, as they have been described above. We assume that all the matrices  $U_k$  have full rank.

Suppose that, starting with  $x_0$ , we have constructed the matrices  $Q_{k-1}$  and  $R_{k-1}$ .

We now read  $x_{k+1}$  and compute  $u_k = x_{k+1} - x_k$ . Following this, using MGS, we compute the scalars  $r_{0k}, r_{1k}, \dots, r_{kk}$  and the orthonormal vector  $q_k$ , which we use to augment the matrices  $Q_{k-1}$  and  $R_{k-1}$  to give  $Q_k$  and  $R_k$ , respectively.

We next proceed to the computation of the  $\gamma_j$ . For MPE, we first solve the upper triangular  $k \times k$  system in (4.14) for  $c_0, c_1, \dots, c_{k-1}$  by back substitution, and then use (2.6) to obtain the  $\gamma_j$ . For RRE, we solve the  $(k+1) \times (k+1)$  system in (4.23) for  $d$ , and then determine the  $\gamma_j$  by (4.21) and (4.22). The solution of the system in (4.23) can be achieved very simply by forward and back substitution as  $R_k$  is



upper triangular.

Once the  $\gamma_j$  have been determined, we compute the  $\xi_j$  by (4.6) and the  $\eta_j$  by (4.9), and finally,  $s_{0,k}$  by (4.8).

Next we read  $x_{k+2}$ , and proceed similarly, until a suitable stopping criterion is met.

It should be noted that, strictly speaking, neither  $r_{kk}$  nor  $q_k$  is needed for determining  $s_{0,k}$ , and their computation can be completed after  $x_{k+2}$  has been introduced. In the computer program that we give in the appendix, though, we chose to compute  $r_{kk}$  and  $q_k$  before the computation of  $s_{0,k}$ .

Finally, it is not difficult to see that these implementations are very appropriate for vector computers as their handling of the  $x_i$ ,  $u_i$ , and  $q_i$  can be entirely vectorized. The computer program given in the appendix to this work has been written to take full account of this.

## 5. ESTIMATION OF RESIDUAL NORMS

### 5.1 General Considerations for Linear and Nonlinear Systems

Let  $s$  be the solution of the linear or nonlinear system of equations

$$x = F(x), \quad (5.1)$$

and let us define the residual for an arbitrary vector  $x$  by

$$r(x) = F(x) - x. \quad (5.2)$$

Let  $x_0$  be a given initial approximation, and generate the sequence of vectors  $x_1, x_2, \dots$ , according to the fixed point iterative method

$$x_{j+1} = F(x_j), \quad j = 0, 1, \dots \quad (5.3)$$

Consequently, the residual for  $x_j$  is given by

$$r(x_j) = F(x_j) - x_j = x_{j+1} - x_j = u_j, \quad (5.4)$$

thus is readily available.

Let us assume that MPE or RRE is applied to the sequence  $x_0, x_1, x_2, \dots$ , and that we are computing the sequence  $s_{n,1}, s_{n,2}, \dots$ . Let us assume also that we would like to stop the computation as soon as some norm of  $r(s_{n,k})$  becomes  $\leq \epsilon$  for some  $k$ ,  $\epsilon > 0$  being a preassigned level of accuracy. The most direct way of doing this would be by actually computing the vectors  $s_{n,1}, r(s_{n,1}), s_{n,2}, r(s_{n,2}), \dots$ , which is very costly.

Indeed, the computation of  $s_{n,k}$  involves about  $k$  vector additions and  $k$  scalar-vector multiplications, that of  $r(s_{n,k})$ , by (5.2), amounts to one additional fixed point iteration and one vector addition, and the computation of the norm of  $r(s_{n,k})$  requires an additional inner product. In addition, the number of the vector operations increases with increasing  $k$ . In view of this, the most desirable situation is one that enables us to estimate some norm of  $r(s_{n,k})$  without having to compute either  $s_{n,k}$  or  $r(s_{n,k})$ .

## 5.2 Residual Computation for Linear Systems

We now devise a strategy by which the  $l_2$ -norms of the residuals  $r(s_{n,k})$  can be obtained exactly without the need to compute either  $s_{n,k}$  or  $r(s_{n,k})$ , when the sequence  $x_0, x_1, x_2, \dots$ , is being generated linearly by the iterative method in (3.2), i.e., when  $F(x) = Ax + b$  in (5.3). The case in which  $F(x)$  is nonlinear will be considered at the end of this section.

When  $F(x) = Ax + b$ , the residual for an arbitrary vector  $x$ , by (5.2), becomes

$$r(x) = Ax + b - x. \quad (5.5)$$

Consequently, by (2.10), (3.2), and (2.1), we have

$$r(s_{0,k}) = \sum_{j=0}^k \gamma_j u_j = U_k \gamma, \quad (5.6)$$

and the  $l_2$ -norm of  $r(s_{0,k})$  is thus

$$\|r(s_{0,k})\| = (r(s_{0,k}), r(s_{0,k}))^{1/2} = (\gamma^* U_k^* U_k \gamma)^{1/2}. \quad (5.7)$$

By invoking  $U_k = Q_k R_k$  in (5.7), we obtain

$$\|r(s_{0,k})\| = (\gamma^* R_k^* R_k \gamma)^{1/2}. \quad (5.8)$$

We now analyze  $\gamma^* R_k^* R_k \gamma$  for MPE and RRE separately.

### 5.2.1 $l_2$ -Norm of Residual with MPE

Let us compute  $R_k \gamma$  first. By (4.12)-(4.14) we have

$$[R_{k-1} \mid \rho_k] \begin{bmatrix} c \\ 1 \end{bmatrix} = 0. \quad (5.9)$$

By dividing both sides of (5.9) by  $\sum_{i=0}^k c_i$  with  $c_k = 1$ , and invoking (2.6), we obtain

$$[R_{k-1} | \rho_k] \gamma = 0. \quad (5.10)$$

Substituting (5.10) in  $R_k \gamma$ , we finally have

$$R_k \gamma = (0, 0, \dots, 0, r_{kk} \gamma_k)^T, \quad (5.11)$$

from which we obtain

$$(\gamma^* R_k^* R_k \gamma)^{1/2} = r_{kk} |\gamma_k|. \quad (5.12)$$

Consequently, for linearly generated sequences

$$\|r(s_{0,k})\| = r_{kk} |\gamma_k| \quad (5.13)$$

exactly, with  $r(x)$  as defined in (5.5).

### 5.2.2 $l_2$ -Norm of Residual with RRE

By (4.19) we have immediately

$$(\gamma^* U_k^* U_k \gamma)^{1/2} = \sqrt{\lambda}, \quad (5.14)$$

with  $\lambda$  as determined from (4.23) and (4.21). Consequently, for linearly generated sequences

$$\|r(s_{0,k})\| = \sqrt{\lambda} \quad (5.15)$$

exactly, with  $r(x)$  as defined in (5.5).

The results given in (5.13) and (5.15) assume exact arithmetic. Due to roundoff errors, however, the actually computed residual norms may be getting farther from (5.13) and (5.15), especially when  $k$  is increasing. In this case it may be appropriate to compute  $s_{0,k}$  and the norm of its residual every once in a while to make sure that roundoff has not started to dominate the computations. Although such a test is not included in the computer program given in the appendix, it is quite easy to incorporate it there.

### 5.3 Practical Residual Estimation in Extrapolation for Nonlinear Systems

We now consider the problem of error estimation for the case in which  $F(x)$  in (5.1) is nonlinear. Let us assume that the sequence  $x_0, x_1, x_2, \dots$ , is convergent, its limit, of course, being  $s$ , the solution of (5.1). Therefore, for  $n$  sufficiently large,  $x_n, x_{n+1}, \dots$ , are all very close to  $s$ , and we have

$$x_{n+1} - s = F'(s)(x_n - s) + \epsilon_n, \quad (5.16)$$

where  $F'(x)$  is the Jacobian matrix of the vector valued function  $F(x)$ , and  $\epsilon_n$  is a vector whose norm

is  $O(\|x_n - s\|^2)$  as  $n \rightarrow \infty$ . This implies that the sequence  $x_0, x_1, x_2, \dots$ , behaves linearly at infinity, in the sense that

$$x_{j+1} \approx F'(s)x_j + (s - F'(s)s) \quad (5.17)$$

for all sufficiently large  $j$ . Thus, for  $n$  sufficiently large, we can take

$$r(s_{n,k}) \approx U_k^{(n)}\gamma \quad (5.18)$$

c.f., (5.6), and

$$\|r(s_{n,k})\| \approx (\gamma^* R_k^{(n)*} R_k^{(n)} \gamma)^{1/2}, \quad (5.19)$$

c.f., (5.8), where we have retained the index  $n$  in  $U_k^{(n)}$  and  $U_k^{(n)} = Q_k^{(n)} R_k^{(n)}$ . The norm in (5.19) is the  $l_2$ -norm as before. Consequently, we can take (5.19) as an estimate for the  $l_2$ -norm of the residual  $r(s_{n,k})$  without having to compute either  $s_{n,k}$  or  $r(s_{n,k})$ , since it is given by (5.12) for MPE and by (5.14) for RRE.

In case  $n$  is not large enough, (5.19) may not be very realistic. In this case we may choose to compute  $s_{n,k}$  and  $r(s_{n,k})$  not for all  $k$ , but for  $k = p, 2p, 3p, \dots$ , say, for some integer  $p > 1$ . This obviously reduces the cost.

When we are using MPE or RRE in the cycling mode, which is one of the best modes of usage, things become simpler. To see this let us recall how cycling can be performed.

Step 1. Fix the integer  $k$ . Pick  $s_k^0 \equiv x_0$  and set  $q = 0$ .

Step 2. Generate  $x_1$ , by (5.3). If  $\|r(s_k^{(q)})\| = \|x_1 - x_0\| = \|u_0\| \leq \epsilon$ , then stop. Otherwise, generate  $x_2, \dots, x_{k+1}$  by (5.3).

Step 3. Compute  $s_k^{(q+1)} \equiv s_{0,k}$  by MPE or RRE.

Step 4. Replace  $x_0$  by  $s_k^{(q+1)}$ , and  $q$  by  $q+1$ , and go to Step 2.

(That  $r(s_k^{(q)}) = u_0$  in Step 2 follows from (5.4).)

Consequently, no extra computation for residuals is necessary, as  $u_0$  is the true residual in each cycle.

## 6. OPERATION COUNT AND STORAGE REQUIREMENTS

In most applications,  $N$ , the dimension of the vectors, is extremely large, while  $k$  takes on very small values. Consequently, the major part of the computational effort is spent in handling the large vectors, the rest being negligible.

As we can easily see, most of the vector computations take place in the QR factorization. At the  $k$ th stage that leads to  $s_{0,k}$ , the vector  $x_{k+1}$  is provided first. Starting with this, we need one vector addition to form  $u_k = x_{k+1} - x_k$ , and, following that,  $k$  vector additions,  $k+1$  scalar-vector multiplications, and  $k+1$  inner products to form the orthonormal vector  $q_k$  and the scalars  $r_{0k}, r_{1k}, \dots, r_{kk}$  by MGS. The computation of  $s_{0,k}$ , if desired, requires  $k$  vector additions and  $k$  scalar-vector multiplications by (4.8). The computation of the  $\gamma_i$ ,  $\xi_i$ , and  $\eta_i$  is negligible, as it involves work with  $k \times k$  or  $(k+1) \times (k+1)$  triangular matrices for very small values of  $k$ .

As for the storage requirements, it is clear that  $x_0$  needs to be saved. At the  $k$ th stage  $q_k$  needs to be saved, in addition to the previously saved  $q_0, q_1, \dots, q_{k-1}$ . We also need two or three more auxiliary vectors of dimension  $N$ . Similarly the elements of the matrix  $R_k$  all need to be saved, but their storage requirements are negligible.

In view of the above, if only  $s_{0,K}$  is needed for some preassigned  $K$ , then, recalling that the vector  $q_K$  need not be computed, the total operation count is  $\frac{1}{2}(K^2 + 5K + 2)$  vector additions,  $\frac{1}{2}(K^2 + 5K)$  scalar-vector multiplications, and  $\frac{1}{2}(K^2 + 3K + 2)$  inner products, which amounts to  $\sim 2K^2N$  floating point operations (scalar additions and multiplications). As for the storage requirements, we need  $(K+1)N$  storage locations for  $x_0, q_0, q_1, \dots, q_{K-1}$ , and  $2N$  storage locations for two additional auxiliary vectors. No additional storage locations are required for  $s_{0,k}$  as  $s_{0,k}$  can overwrite  $x_0$  at the end of the computation.

In many cases it turns out that the accuracy that can be achieved with  $m$  cycles of MPE or RRE, each cycle being of width  $K$ , is comparable to that obtained for  $s_{0,mK}$ . If we compare the computational costs of each of these strategies, we see that, roughly speaking, the former is  $m$  times less expensive computationally than the latter, and requires  $m$  times less storage. Thus, as a computational strategy, cycling possesses important advantages.

It is very instructive to compare the implementations for MPE and RRE, as they are given in this work, with the vector epsilon algorithm (VEA) of Wynn [21]. VEA is defined recursively by

$$\begin{aligned} \varepsilon_{-1}^{(n)} &= 0 \quad \text{and} \quad \varepsilon_0^{(n)} = x_n, \quad n = 0, 1, \dots, \\ \varepsilon_{k+1}^{(n)} &= \varepsilon_k^{(n+1)} + \frac{\overline{\Delta \varepsilon_k^{(n)}}}{(\Delta \varepsilon_k^{(n)}, \Delta \varepsilon_k^{(n)})}, \quad k \geq 0, \quad n \geq 0, \end{aligned} \quad (6.1)$$

where  $\Delta \varepsilon_k^{(n)} = \varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}$ , and  $\bar{z} = (\bar{z}_1, \dots, \bar{z}_N)^T$  if  $z = (z_1, \dots, z_N)^T$ . Thus, the computation of  $\varepsilon_k^{(n)}$  for  $k \geq 2$  requires two vector additions, one scalar-vector multiplication, and one inner product. For  $\varepsilon_1^{(n)}$  only one vector addition is required. Now as is suggested by experience and as can be justified heuristically, for given  $K$ ,  $\varepsilon_{2K}^{(0)}$  for VEA and  $s_{0,K}$  for MPE or RRE would have comparable performance. The total operation count for determining  $\varepsilon_{2K}^{(0)}$  is  $4K^2$  vector additions,  $2K^2 + K$  scalar-vector multiplications, and  $2K^2 + K$  inner products, which amounts to  $\sim 10K^2N$  floating points operations (scalar additions and multiplications). As for the storage requirements, we need  $(2K+1)N$  storage locations to save  $\varepsilon_0^{(2K)}, \varepsilon_1^{(2K-1)}, \dots, \varepsilon_{2K}^{(0)}$ , and  $2N$  storage locations for two auxiliary vectors. Consequently, VEA is about five times more expensive than either MPE or RRE as far as operation counts are concerned. As far as storage requirements are concerned, VEA is about twice as expensive as either MPE or RRE. In addition, since  $x_0, x_1, \dots, x_{2K}$  are needed for  $\varepsilon_{2K}^{(0)}$ , whereas, only  $x_0, x_1, \dots, x_{K+1}$  are needed for either MPE or RRE, VEA is about twice as expensive as MPE or RRE with respect to the number of vectors they utilize.

We note that, in the epsilon family of vector extrapolation methods, VEA seems to be the most advantageous as far as the operation count, storage requirements, and numerical stability are concerned. For more details, see [19].

## 7. SOME PRACTICAL CONSIDERATIONS FOR ENHANCING CONVERGENCE AND STABILITY

In this section we would like to make a few remarks, which we believe are of practical importance with regard to enhancing the convergence and stability of vector extrapolation methods as they are applied to iterative procedures. Most of these remarks are based on the known theoretical results concerning vector extrapolation methods, some of which have been discussed in Section 3.

### 7.1 Effect of Iteration Before Extrapolation

In most problems of interest the vector sequence  $x_0, x_1, \dots$ , converges extremely slowly so that there is not much difference between  $\|x_n - s\|$  and  $\|x_0 - s\|$  even for appreciably large values of  $n$ . The result in (3.6), however, suggests that there may be a large difference between  $\|s_{n,k} - s\|$  and

$\|x_n - s\|$  (hence  $\|x_0 - s\|$ ) if  $n$  is sufficiently large. If the vectors  $x_j$  are produced by an iterative procedure such as (3.2), then this implies that it may be very useful to start the extrapolation procedure after a number of iterations with (3.2). One heuristic argument in favor of this strategy runs as follows: The initial error  $x_0 - s$ , in general, has components in the direction of all eigenvectors and principal vectors of  $A$ . After a few iterations the components in the direction of those eigenvectors and principal vectors corresponding to zero eigenvalues of  $A$  are totally eliminated, while those corresponding to the eigenvalues that are close to zero are diminished. Consequently, the error vector  $x_n - s$  has mostly contributions from the eigenvectors and principal vectors corresponding to the large eigenvalues. Precisely these contributions are now diminished by the extrapolation procedure.

## 7.2 A Simple "Averaging" of the Iteration Process and Its Effect on Convergence and Stability

Assume that (3.2) or (5.3) result from the discrete solutions of continuum problems. Then, for a convergent scheme, the largest eigenvalues of  $A$  or of  $F'(s)$ , the Jacobian matrix of  $F(x)$  at  $x = s$ , may be very close to 1 in the complex plane in some cases. This may cause the extrapolation process not to be very effective. The process may even suffer from a large amount of numerical instability.

One way of dealing with this problem is by applying extrapolation methods not to the sequence  $x_0, x_1, x_2, \dots$ , but to  $y_0, y_1, y_2, \dots$ , where  $y_j = x_{jp}$ , for some positive integer  $p$ . This strategy has been successfully implemented in [17].

Another way would be by changing (5.3), in general, to read

$$x_{j+1} = x_j + \omega(F(x_j) - x_j) = (1-\omega)x_j + \omega F(x_j), \quad j = 0, 1, \dots, \quad (7.1)$$

where  $\omega$  is a scalar different than 1. (The sequence generated by taking  $\omega = 1$  is the one generated by (5.3).) Thus  $x_{j+1}$  is now a weighted "average" of  $x_j$  and  $F(x_j)$ , in which the weights  $1-\omega$  and  $\omega$  need not be both positive.

By picking  $\omega$  appropriately we can cause the spectrum of the Jacobian matrix of  $(1-\omega)x + \omega F(x)$  at  $x = s$ , namely,  $(1-\omega)I + \omega F'(s)$ , to be increasingly favorable to  $s_{n,k}$  for large values of  $n$ .

Let us take a look at the following example: Suppose the eigenvalues of  $F'(s)$  are all positive and lie in the interval  $[\epsilon, 1-\eta]$  for some  $\epsilon > 0$  and  $\eta > 0$  close to zero. Consequently, the sequence  $x_0, x_1, \dots$ , obtained from (5.3) converges, provided  $x_0$  is sufficiently close to  $s$  in case  $F(x)$  is non-linear, and unconditionally in case  $F(x)$  is linear. If we pick  $\omega = 2$ , then the eigenvalues of

$(1-\omega)I + \omega F'(s)$  lie in the interval  $[-1+2\varepsilon, 1-2\eta]$  so that the sequence obtained from (7.1) also converges. (If  $\varepsilon = \eta$ , then this sequence converges more quickly than the one obtained from (5.3).) The new spectrum has two important properties relevant to vector extrapolation methods: 1) The largest positive eigenvalue of  $F'(s)$ , namely,  $1-\eta$ , has moved away from 1. 2) Negative eigenvalues close to  $-1$  have been created. Both of these properties enhance the stability of vector extrapolation processes both mathematically and numerically. (This follows from [12, Theorem 4.1], [16, Theorem 3.2], and [18, Theorems 4.1 and 5.2].) It should be noted that 2 is also that value of  $\omega$  for which the spectral radius of  $(1-\omega)I + \omega F'(s)$  is minimal when  $\varepsilon = \eta$ .

### 7.2.1 Special Considerations for Linear Systems

When  $F(x) = Ax + b$ , and the vector sequence is generated by the iterative procedure in (7.1), the approximations  $s_{0,k}$  are independent of  $\omega$ , as has been shown by Israeli and Sidi [8]. That is to say, the convergence properties of the  $s_{0,k}$  are not changed by varying  $\omega$ . Nevertheless, varying  $\omega$  may influence the stability properties of the numerical implementations.

First, if the sequence obtained from (3.2) is divergent, then all the computations leading to  $s_{0,k}$  will suffer a large loss of accuracy, especially for increasing  $k$ . By changing  $\omega$  in (7.1) appropriately, we can cause the sequence to converge (or diverge very slowly), thus avoiding the numerical problem caused by the unboundedness of the original sequence.

Next, if the sequence obtained from (3.2) is slowly converging on account of the largest eigenvalues of  $A$  all being very close to 1 in the complex plane, then the vectors  $u_0, u_1, u_2, \dots$ , are near being linearly dependent. Consequently, the  $l_2$  condition number of the matrices  $U_k$  may be very large. This may have a negative influence on the QR factorization of  $U_k$  by MGS that we have chosen for our implementation. This influence exhibits itself in the computed matrices  $Q_k$  being far from unitary and the computed  $s_{0,k}$  not being very accurate. If, by picking  $\omega$  appropriately in (7.1), we can change the spectrum in such a way that it now contains both positive and negative large eigenvalues, then the vectors  $u_0, u_1, u_2, \dots$ , will be far from being linearly dependent numerically. This will result in better conditioned matrices  $U_k$ , which, in turn, will result in the computed matrices  $Q_k$  being closer to unitary and the computed  $s_{0,k}$  being quite accurate.

The numerical aspects of MGS and its use in the solution of least squares problems and the comparison of these with the Householder QR factorization and least squares solutions are discussed at



length in [7, Sections 5.2.8, 5.2.9, and 5.3.6].

### 7.2.2 Application to Jacobi Iteration for Consistently Ordered Matrices

The observations above can be used very effectively in the solution of linear systems whose matrices are consistently ordered. Such matrices arise frequently, for example, in the finite difference solutions of elliptic equations.

Suppose iterative methods of the form (3.2) are being used in the solution of such a system. If the method used is the Jacobi iteration method, then it is known that the nonzero eigenvalues of  $A$  come in pairs of the form  $\pm \mu$ , see, e.g., Varga [20, Chapter 4]. Consequently, if the eigenvalues of  $A$  are real, then they are in the interval  $[-1+\delta, 1-\delta]$  for some  $\delta$ ,  $0 < \delta < 1$ , provided  $\rho(A) < 1$ . As a result, the nonzero eigenvalues of  $A^2$  are in the interval  $[\epsilon, 1-\eta]$ , for some  $\epsilon > 0$ , where  $1-\eta = (1-\delta)^2 = 1-2\delta$  if  $\delta \ll 1$ . Furthermore, if  $2M$  is the number of the distinct nonzero eigenvalues of  $A$ , then the number of the distinct nonzero eigenvalues of  $A^2$  is  $M$  whether the eigenvalues of  $A$  are real or not.

This implies that the approximation  $s_{2n, 2k}^1$  obtained from the Jacobi iterative method and the approximation  $s_{n, k}^2$  obtained from the double Jacobi iterative method

$$\begin{aligned} y &= Ax_j + b \\ x_{j+1} &= Ay + b, \quad j = 0, 1, \dots, \end{aligned} \tag{7.2}$$

have the same asymptotic behavior as  $n \rightarrow \infty$ . In addition, since the largest eigenvalues of  $A^2$  are twice as far from 1 as those of  $A$ ,  $s_{n, k}^2$  is more stable than  $s_{2n, 2k}^1$  as  $n \rightarrow \infty$  both mathematically and numerically.

We can now couple the double Jacobi iteration method with the simple averaging procedure that was discussed above. The new iteration procedure then is

$$\begin{aligned} y &= Ax_j + b \\ z &= Ay + b \\ x_{j+1} &= (1-\omega)x_j + \omega z, \quad j = 0, 1, \dots, \end{aligned} \tag{7.3}$$

for some  $\omega \neq 0$ . As explained before, by varying  $\omega$  we can cause the spectrum of the iteration matrix of (7.3), namely,  $(1-\omega)I + \omega A^2$ , to become favorable to  $s_{n, k}$ . In particular, by picking  $\omega = 2$  we can cause this spectrum to lie in the interval  $[-1+2\epsilon, 1-2\eta] = [-1+2\epsilon, 1-4\delta+\delta^2]$ . This enlarges the distance of the largest positive eigenvalue of the Jacobi iteration matrix  $A$  from 1 even further, and introduces

negative eigenvalues close to  $-1$ . This causes  $s_{n,k}^2$  to become more stable. Furthermore, if  $\epsilon \geq \delta$ , the convergence rate of  $x_n$  from (7.3) with  $\omega = 2$  is as good as that of  $x_n$  from (7.2).

We note, incidentally, that the iterative method of (7.3) with  $\omega = 2$  is known as Abramov's method, see Faddeev and Faddeeva [5, p. 514]. It is quite easy to see that, in this case,

$$x_{j+1}-s = (2A^2-I)(x_j-s) = T_2(A)(x_j-s),$$

where  $T_2(\lambda) = 2\lambda^2 - 1$  is the Chebyshev polynomial of degree two. It should be emphasized that this is not Chebyshev acceleration, however.

## 8. NUMERICAL EXAMPLES

We have applied MPE and RRE through their new implementations described in the previous sections to several examples. This has been done by employing the computer program that is provided in Appendix B of this work. Some of the results obtained this way will be reported in this section.

We have picked real linear systems of equations whose matrices are symmetric or nonsymmetric. Numerical results for two of these systems, one symmetric and the other nonsymmetric, are included in this work.

**Example 1.** Consider the vector sequence obtained from (3.2), where  $A$  is a  $1000 \times 1000$  septadiagonal matrix symmetric with respect to both of its main diagonals, and is given by

$$A = 0.06 \times \begin{bmatrix} 5 & 2 & 1 & 1 & & & \\ 2 & 6 & 3 & 1 & 1 & & \\ 1 & 3 & 6 & 3 & 1 & 1 & \\ 1 & 1 & 3 & 6 & 3 & 1 & 1 \\ & 1 & 1 & 3 & 6 & 3 & 1 & 1 \\ & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}.$$

The vector  $b$  is such that the exact solution  $s$  of (3.1) is  $(1, 1, \dots, 1)^T$ .

All eigenvalues of  $A$  are in  $(0, 1)$ , the smallest and the largest being  $4.7279 \dots \times 10^{-6}$  and  $0.95999 \dots$ , respectively. Consequently, the matrix  $C = I - A$  is symmetric positive definite. Also, there is a large amount of clustering of eigenvalues near the smallest and the largest ones.

Taking  $x_0 = 0$ , we generated the vectors  $x_1, x_2, \dots$ , by (7.1) once by taking  $\omega = 1$  and once by taking  $\omega = 2$ , and then applied MPE to these two sequences. We also applied the method of conjugate gradients (CG) to the linear system  $Cx = b$  starting again with  $x_0 = 0$ . The results of these computations are shown in Table 1a.

Recall that the Arnoldi method becomes equivalent to CG when  $C$  is a symmetric matrix, and MPE, when applied to a linearly generated sequence, becomes equivalent to the Arnoldi method. Also  $s_{0,k}$ , when applied to a sequence generated linearly as in (7.1), is independent of  $\omega$ . Consequently,  $s_{0,k}$ , both for  $\omega = 1$  and  $\omega = 2$ , obtained from MPE, and  $z_k$ , obtained from CG, are all the same mathematically. This is verified in Table 1a at least for  $k \leq 10$ . The differences between the  $\omega = 1$  and  $\omega = 2$  MPE computations for  $k > 10$  can be explained exactly as described at the end of Section 7.2.1. Again, as can be seen from Table 1a, the  $\omega = 2$  MPE computation differs from the CG computation starting with  $k = 40$  approximately. Since CG involves orthogonalization with respect to only one vector, its absolute accuracy is guaranteed. On the other hand, MPE involves orthogonalization with respect to an ever increasing number of vectors at each stage, thus it cannot be absolutely accurate. In spite of this, the present implementation of MPE seems to be very stable in the sense that  $\|s_{0,k} - s\|$  seems to be constantly decreasing with increasing  $k$ . Indeed, we have verified this by going up to  $k = 100$  in both the  $\omega = 1$  and  $\omega = 2$  MPE computations.

Our purpose in presenting Table 1a was to demonstrate the good stability properties of the new MPE implementation for large values of  $k$ . Otherwise, CG is the method we would normally use for this example, since its operation count and storage requirements are extremely small.

In Table 1b we present the results obtained for the same example with  $\omega = 2$  first performing 20 iterations and then using MPE in the cycling mode with  $k = 10$ , as explained at the end of Section 5. The remarkable effectiveness of this strategy is obvious.

$k$	MPE						CG
	$\omega = 1$			$\omega = 2$			$\ z_k - s\ $
	$r_{kk}  \gamma_k $	$\ r(s_{0,k})\ $	$\ s_{0,k} - s\ $	$r_{kk}  \gamma_k $	$\ r(s_{0,k})\ $	$\ s_{0,k} - s\ $	
0	1.46D+00	1.46D+00	3.16D+01	2.92D+00	2.92D+00	3.16D+01	3.16D+01
5	1.92D-01	1.92D-01	1.17D+00	3.83D-01	3.83D-01	1.17D+00	1.17D+00
10	1.98D-02	1.98D-02	1.53D-01	3.96D-02	3.96D-02	1.53D-01	1.53D-01
15	2.51D-03	2.51D-03	2.03D-02	5.01D-03	5.01D-03	2.02D-02	2.02D-02
20	4.51D-04	4.53D-04	3.70D-03	6.63D-04	6.63D-04	2.68D-03	2.68D-03
25	1.58D-04	2.26D-04	4.43D-03	8.78D-05	8.78D-05	3.52D-04	3.52D-04
30	6.27D-05	1.11D-04	2.44D-03	1.15D-05	1.15D-05	4.63D-05	4.63D-05
35	2.49D-05	3.19D-05	6.08D-04	1.53D-06	1.53D-06	6.53D-06	6.11D-06
40	6.37D-06	1.42D-05	3.34D-04	5.16D-07	5.30D-07	1.64D-06	8.03D-07
45	7.93D-06	5.42D-06	3.66D-05	7.31D-08	1.29D-07	1.27D-06	1.06D-07
50	9.49D-07	1.29D-06	1.36D-04	3.17D-08	4.29D-08	1.85D-07	1.19D-08

**Table 1a** - Numerical results for Example 1, starting with  $x_0 = 0$ .

$i$	$\ r(s_k^{(i)})\ $	$\ s_k^{(i)} - s\ $
0	4.75 D-01	5.91 D+00
1	2.00 D-04	6.94 D-04
2	2.90 D-06	8.78 D-06
3	4.17 D-08	1.74 D-07
4	9.27 D-10	3.70 D-09
5	2.18 D-11	9.11 D-11
6	5.49 D-13	2.83 D-12
7	4.26 D-14	1.77 D-13
8	6.16 D-15	9.46 D-14

**Table 1b** - MPE applied to Example 1 in the cycling mode. Starting with the zero vector, first 20 iterations are performed. Following that MPE is applied in the cycling mode with  $k = 10$ . The  $l_2$ -norm of the error in the initial (zero) vector is 3.16D+01. The vectors are obtained by "averaging" the iterative process (3.2) with  $\omega = 2$ .

**Example 2.** Consider the linear nonsymmetric system of equations  $\tilde{C}x = \tilde{b}$ , where  $\tilde{C}$  is the block - tri-diagonal matrix

$$\tilde{C} = \begin{bmatrix} B & -I & & & \\ -I & B & -I & & \\ & -I & B & -I & \\ & & & \ddots & \\ & & & & -I & B \end{bmatrix} \quad \text{with } B = \begin{bmatrix} 4 & & & & \\ & a & & & \\ b & & & & \\ & & & & a \\ & & & & b & 4 \end{bmatrix},$$

and  $a = -1 + \delta$ ,  $b = -1 - \delta$ ,  $\delta \neq 0$ . (See [10, p. 122].) Again, the vector  $\tilde{b}$  is such that the exact solution  $s$  is  $(1, 1, \dots, 1)^T$ . The iterative method that we pick for this system is Jacobi's method, so that  $A = I - \frac{1}{4}\tilde{C}$ .

Now the matrix  $\tilde{C}$  is consistently ordered. Thus the suggestions put forth in Section 7.2.2 can be successfully employed in this case.

In our numerical experiments we took  $\delta = 0.2$ . The matrices  $B$  and  $I$  in  $\tilde{C}$  were all  $10 \times 10$  and  $\tilde{C}$  was  $200 \times 200$ , exactly as in [10]. The extrapolation method for which we give numerical results is RRE. We first applied RRE in the cycling mode in conjunction with Jacobi iteration. The vector obtained at the end of each cycle is denoted  $\bar{s}_k^{(i)}$ . Next we applied RRE in the cycling mode in conjunction with double Jacobi iteration. The vector obtained at the end of each cycle is denoted  $\hat{s}_k^{(i)}$  now. Finally, we applied RRE in the cycling mode extended as follows: The vector sequence is generated by the iterative procedure of (7.3) with  $\omega = 2$ , i.e., by the "averaged" double Jacobi iteration with  $\omega = 2$ . In each cycle  $n_i + k_i + 1$  such iterations are performed, and extrapolation is applied to the last  $k_i + 2$  of the vectors, i.e., in each cycle  $s_{n_i, k_i}$  is computed. The vector obtained at the end of each cycle now is denoted  $\tilde{s}_{n_i, k_i}^{(i)}$ . The index  $i$  denotes the cycle number in each case.

In Table 2 we give the  $l_2$  - norms of the errors  $\bar{s}_k^{(i)} - s$  ( $k = 20$ ),  $\hat{s}_k^{(i)} - s$  ( $k = 10$ ), and  $\tilde{s}_{n_i, k_i}^{(i)}$  ( $n_i = 5$ ,  $k_i = 5$  all  $i$ ). Thus the number of basic Jacobi iterations performed to obtain the approximations  $\bar{s}_k^{(i)}$ ,  $\hat{s}_k^{(i)}$ , and  $\tilde{s}_{n_i, k_i}^{(i)}$  in each cycle is 21, 22, and 22 respectively. We see that  $\bar{s}_{20}^{(i)}$  and  $\hat{s}_{10}^{(i)}$  have comparable accuracy, as expected. The number of vector operations for  $\tilde{s}_{20}^{(i)}$ , however, is over three

times that for  $\hat{s}_{10}^{(i)}$ . Also the storage requirement for  $\bar{s}_{10}^{(i)}$  is about twice that for  $\hat{s}_{10}^{(i)}$ . The performance of  $\bar{s}_{5,5}^{(i)}$  is only slightly inferior. The number of vector operations for  $\bar{s}_{5,5}^{(i)}$  is about one tenth that for  $\bar{s}_{20}^{(i)}$ , while its storage needs are about one third those of  $\bar{s}_{20}^{(i)}$ .

$i$	$\ \bar{s}_{20}^{(i)} - s\ $	$\ \hat{s}_{10}^{(i)} - s\ $	$\ \bar{s}_{5,5}^{(i)} - s\ $
1	6.66D-02	7.47D-02	1.34D-01
2	2.02D-04	2.36D-04	5.86D-04
3	2.53D-07	4.26D-07	1.14D-05
4	2.90D-10	2.05D-09	3.04D-08
5	2.03D-12	5.96D-12	2.15D-10
6	1.35D-13	6.48D-14	1.07D-12
7	3.61D-14	3.13D-14	1.75D-14

Table 2 - RRE applied to Example 2 in the cycling mode. The initial vector is zero, and the  $l_2$ -norm of the error associated with it is 1.41D+01.

## ACKNOWLEDGEMENT

The author gratefully acknowledges the useful conversations that he has had with Prof. Moshe Israeli in the course of this work. The computations reported in this paper were done in double precision arithmetic on an IBM-370 computer at the Computation Center, Technion - Israel Institute of Technology.

## REFERENCES

- [1] O. Axelsson, "Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations", *Lin. Alg. Appl.*, **29** (1980), pp. 1-16.
- [2] S. Cabay and L.W. Jackson, "A polynomial extrapolation method for finding limits and antilimits of vector sequences", *SIAM J. Numer. Anal.*, **13** (1976), pp. 734-752.
- [3] R.P. Eddy, "Extrapolating to the limit of a vector sequence", in *Information Linkage between Applied Mathematics and Industry*, (P.C.C. Wang, ed.), Academic Press, New York, 1979, pp. 387-396.
- [4] S.C. Eisenstat, H.C. Elman, and M. Schultz, "Variational iterative methods for nonsymmetric systems of linear equations", *SIAM J. Numer. Anal.*, **20** (1983), pp. 345-357.
- [5] D.K. Faddeev and V.N. Faddeeva, *Computational Methods of Linear Algebra*, W.H. Freeman and Co., San Francisco, 1963.
- [6] W.F. Ford and A. Sidi, "Recursive algorithms for vector extrapolation methods", *Appl. Num. Math.*, **4** (1988), pp. 477-489.
- [7] G.H. Golub and C.F. Van Loan, *Matrix Computations*, Second Edition, Johns Hopkins University Press, Baltimore, 1989.
- [8] M. Israeli and A. Sidi, "An invariance property of some vector extrapolation methods". In preparation.
- [9] M. Mešina, "Convergence acceleration for the iterative solution of the equations  $X=AX+f$ ", *Comp. Meth. Appl. Mech. Eng.*, **10** (1977), pp. 165-173.
- [10] Y. Saad, "Krylov subspace methods for solving large unsymmetric linear systems", *Math. Comp.*, **37** (1981), pp. 105-126.
- [11] Y. Saad and M.H. Schultz, "A generalized minimal residual algorithm for solving nonsymmetric linear systems", *SIAM J. Sci. Stat. Computing*, **7** (1986), pp. 856-869.
- [12] A. Sidi, "Convergence and stability properties of minimal polynomial and reduced rank extrapolation algorithms", *SIAM J. Numer. Anal.*, **23** (1986), pp. 197-209. Originally appeared as NASA TM-83443 (July 1983).
- [13] A. Sidi, "Extrapolation vs. projection methods for linear systems of equations", *J. Comp. Appl. Math.*, **22** (1988), pp. 71-88.
- [14] A. Sidi, "On extensions of the power method for normal operators", *Lin. Alg. Appl.*, **120** (1989), pp. 207-224.
- [15] A. Sidi, "Application of vector extrapolation methods to consistent singular systems". TR#540, Computer Science Dept., Technion, Haifa, Israel (February 1989). To appear in *Applied Numerical Mathematics*.
- [16] A. Sidi and J. Bridger, "Convergence and stability analyses for some vector extrapolation methods in the presence of defective iteration matrices", *J. Comp. Appl. Math.*, **22** (1988), pp. 35-61.
- [17] A. Sidi and M.L. Celestina, "Convergence acceleration for vector sequences and applications to computational fluid dynamics". NASA TM-101327, ICOMP-88-17 (August 1988).
- [18] A. Sidi, W.F. Ford, and D.A. Smith, "Acceleration of convergence of vector sequences", *SIAM J. Numer. Anal.*, **23** (1986), pp. 178-196. Originally appeared as NASA TP-2193 (December 1983).
- [19] D.A. Smith, W.F. Ford, and A. Sidi, "Extrapolation methods for vector sequences", *SIAM Rev.*, **29** (1987), pp. 199-233. Erratum: Correction to "Extrapolation methods for vector sequences", *SIAM Rev.*, **30** (1988), pp. 623-624.
- [20] R.S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, New York, 1962.
- [21] P. Wynn, "Acceleration techniques for iterated vector and matrix problems", *Math. Comp.*, **16** (1962), pp. 301-322.
- [22] D.M. Young and K.C. Jea, "Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods", *Lin. Alg. Appl.*, **34** (1980), pp. 159-194.

## APPENDIX A

**Lemma:** Let  $T$  be an  $m \times m$  hermitian positive definite matrix, and let  $z_1, z_2, \dots, z_m$  be complex variables. Denote  $z = (z_1, z_2, \dots, z_m)^T$ . Then the solution to the problem

$$\begin{aligned} & \text{minimize } z^* T z \\ & \text{subject to } \sum_{i=1}^m z_i = 1 \end{aligned} \quad (\text{A.1})$$

can be obtained by solving the linear system of  $m+1$  equations

$$\begin{aligned} T z &= \lambda \tilde{e} \\ \sum_{i=1}^m z_i &= 1, \end{aligned} \quad (\text{A.2})$$

where  $z_1, \dots, z_m$ , and  $\lambda$  are unknowns, and

$$\tilde{e} = (1, 1, \dots, 1)^T. \quad (\text{A.3})$$

The unknown  $\lambda$  turns out to be real and positive, and is given by

$$\lambda = z^* T z \quad \text{at the solution.} \quad (\text{A.4})$$

The solution of (A.2) can be achieved by first solving the system

$$T h = \tilde{e} \quad (\text{A.5})$$

for  $h = (h_1, \dots, h_m)^T$ , and letting

$$\lambda = \left( \sum_{i=1}^m h_i \right)^{-1}, \quad (\text{A.6})$$

and finally setting

$$z = \lambda h. \quad (\text{A.7})$$

**Proof:** We start by expressing the problem in terms of real variables. Let us write  $T$  in the form

$$T = M + iN, \quad M \text{ and } N \text{ real } m \times m \text{ matrices.} \quad (\text{A.8})$$

Then, by the assumption that  $T$  is hermitian, it follows that

$$M^T = M \quad \text{and} \quad N^T = -N. \quad (\text{A.9})$$

Writing

$$z = x + iy, \quad x \text{ and } y \text{ real } m\text{-dimensional vectors,} \quad (\text{A.10})$$



and invoking (A.8) and (A.9) in  $z^* T z$ , we have

$$z^* T z = x^T M x + y^T M y + 2y^T N x . \quad (\text{A.11})$$

Now the constraint  $\sum_{i=1}^m z_i = 1$  in (A.1) is equivalent to two real constraints, namely,

$$\sum_{i=1}^m x_i = 1 \quad \text{and} \quad \sum_{i=1}^m y_i = 0 . \quad (\text{A.12})$$

We now use the method of Lagrange multipliers to minimize (A.11) subject to (A.12). Introducing the Lagrange multipliers  $-2\mu$  and  $-2\nu$  for the constraints  $\sum_{i=1}^m x_i = 1$  and  $\sum_{i=1}^m y_i = 0$ , respectively, and taking derivatives with respect to the  $x_i$  and  $y_i$ , we obtain the linear system of equations

$$\begin{aligned} Mx - Ny - \mu \tilde{e} &= 0 \\ My + Nx - \nu \tilde{e} &= 0, \end{aligned} \quad (\text{A.13})$$

which, upon letting  $\lambda = \mu + iv$ , becomes equivalent to  $Tz = \lambda \tilde{e}$ . We have thus shown the truth of (A.2). Multiplying  $Tz = \lambda \tilde{e}$  on the left by  $z^*$ , and using  $\sum_{i=1}^m z_i = 1$ , we obtain (A.4). Obviously,  $\lambda$  has to be strictly positive. For if  $\lambda$  were zero, then  $z=0$  would have to be the solution as  $T$  is hermitian positive definite, but this would contradict the constraint  $\sum_{i=1}^m z_i = 1$ . The rest of the proof follows easily from (A.2), and we shall omit it. □

## APPENDIX B

In this appendix we give a computer code written in standard FORTRAN 77 that implement MPE and RRE as described in the present work.

The implementation of MPE and RRE is done in SUBROUTINE MPPERRE that forms the heart of this code.

Use of MPE and RRE in the cycling mode is made possible by SUBROUTINE CYCLE.

The vector sequence for extrapolation is generated by calling SUBROUTINE VECTOR, which, in the present code provides the iteration sequence of Example 1 with  $\omega = 2$  weighting.

The driving program in the present code is the one that generates some of results shown in Table 1b.

We give no further explanations about the code and its use, as the different parts of the code are documented in detail.

```

C*****CYC00010
C IMPLEMENTATION OF MPE AND RRE WITH QR FACTORIZATION FOR LEAST CYC00020
C SQUARES. (QR PERFORMED BY MODIFIED GRAM-SCHMIDT PROCESS) CYC00030
C MPE AND RRE ARE APPLIED IN THE CYCLING MODE. CYC00040
C*****CYC00050
C THE COMPONENTS OF THE INITIAL VECTOR X, NAMELY, X(I), I=1,...,NDIM, CYC00060
C CAN BE PICKED RANDOMLY. WE ACHIEVE THIS, E.G., BY INVOKING THE CYC00070
C IMSL VERSION 10 SUBROUTINE DRNUN THAT GENERATES PSEUDORANDOM CYC00080
C NUMBERS FROM A UNIFORM (0,1) DISTRIBUTION. CYC00090
C OTHER CHOICES FOR X(1),...,X(NDIM) ARE POSSIBLE, SUCH AS X(I)=0, CYC00100
C I=1,...,NDIM. IN THIS CASE REPLACE THE STATEMENT CYC00110
C CALL DRNUN(NDIM,X) CYC00120
C BY THE DO LOOP CYC00130
C DO 10 I=1,NDIM CYC00140
C X(I)=0 CYC00150
C 10 CONTINUE CYC00160
C*****CYC00170
C IMPLICIT DOUBLE PRECISION (A-H,O-Z) CYC00180
C PARAMETER (METHOD=1,N0=20,N=0,KMAX=10,NCYCLE=15,NDIM=1000) CYC00190
C PARAMETER (EPSC=1D-10,IPRES=1,IPRES1=1) CYC00200
C DIMENSION X(NDIM),S(NDIM),Y(NDIM),Z(NDIM) CYC00210
C DIMENSION Q(NDIM,0:KMAX-1),R(0:KMAX,0:KMAX) CYC00220
C DIMENSION C(0:KMAX),GAMMA(0:KMAX),XI(0:KMAX-1) CYC00230
C EXTERNAL VECTOR CYC00240
C CYC00250
C INITIAL VECTOR DETERMINATION. CYC00260
C CYC00270
C CALL DRNUN(NDIM,X) CYC00280
C DO 10 I=1,NDIM CYC00290
C X(I)=0 CYC00300
C 10 CONTINUE CYC00310
C CYC00320
C END OF INITIAL VECTOR DETERMINATION. CYC00330
C CYC00340
C CALL CYCLE(METHOD,X,S,N0,N,KMAX,NCYCLE,NDIM,Y,Z,VECTOR,Q,R, CYC00350
C *C,GAMMA,XI,RESC,EPSC,IPRES,IPRES1) CYC00360
C STOP CYC00370
C END CYC00380
C CYC00390
C SUBROUTINE CYCLE(METHOD,X,S,N0,N,KMAX,NCYCLE,NDIM,Y,Z,VECTOR,Q,R, CYC00400
C *C,GAMMA,XI,RESC,EPSC,IPRES,IPRES1) CYC00410
C*****CYC00420
C THIS SUBROUTINE APPLIES MPE AND RRE IN THE CYCLING MODE. CYC00430
C MPE AND RRE ARE INVOKED BY CALLING SUBROUTINE MPERRE. CYC00440
C*****CYC00450
C THE ARGUMENTS METHOD,NDIM,Y,Z,VECTOR,Q,R,C,GAMMA,XI,IPRES,IPRES1 CYC00460
C ARE AS IN SUBROUTINE MPERRE. CYC00470
C CYC00480
C X : INITIAL VECTOR. INPUT ARRAY OF DIMENSION NDIM. (DOUBLE CYC00490
C PRECISION) CYC00500
C S : THE FINAL APPROXIMATION PRODUCED BY THE SUBROUTINE. OUTPUT CYC00510
C ARRAY OF DIMENSION NDIM. (DOUBLE PRECISION) CYC00520
C N0 : NUMBER OF ITERATIONS PERFORMED BEFORE CYCLING IS STARTED, CYC00530
C I.E., BEFORE MPE OR RRE IS APPLIED FOR THE FIRST TIME. CYC00540
C INPUT. (INTEGER) CYC00550
C N : NUMBER OF ITERATIONS PERFORMED BEFORE MPE OR RRE IS APPLIED CYC00560
C IN EACH CYCLE AFTER THE FIRST CYCLE. INPUT. (INTEGER) CYC00570
C KMAX : WIDTH OF EXTRAPOLATION. ON EXIT FROM SUBROUTINE MPERRE IN CYC00580
C EACH CYCLE, THE ARRAY S IS, IN FACT, THE APPROXIMATION CYC00590
C S(N0,KMAX) IN THE FIRST CYCLE,AND S(N,KMAX) IN THE FOLLOWING CYC00600
C CYCLES. INPUT. (INTEGER) CYC00610
C NCYCLE: MAXIMUM NUMBER OF CYCLES ALLOWED. INPUT. (INTEGER) CYC00620
C RESC : L2-NORM OF THE RESIDUAL FOR S AT THE END OF EACH CYCLE. CYC00630
C RETRIEVED AT THE END OF THE NEXT CYCLE. OUTPUT. (DOUBLE CYC00640
C PRECISION) CYC00650
C EPSC : AN UPPER BOUND ON RESC/RESP, SOME RELATIVE RESIDUAL FOR S, CYC00660

```

```

C      USED IN THE STOPPING CRITERION. HERE RESP IS THE L2-NORM      CYC00670
C      OF THE RESIDUAL FOR S(N0,KMAX) AT THE END OF THE FIRST      CYC00680
C      CYCLE, I.E., ON EXIT FROM SUBROUTINE MPPERRE THE FIRST TIME. CYC00690
C      IF RESC.LE.EPSC*RESP AT THE END OF SOME CYCLE, THEN ONE     CYC00700
C      ADDITIONAL CYCLE IS PERFORMED, AND THE CORRESPONDING        CYC00710
C      S(N,KMAX) IS ACCEPTED AS THE FINAL APPROXIMATION, AND THE    CYC00720
C      SUBROUTINE IS EXITED. INPUT. (DOUBLE PRECISION)             CYC00730
C*****CYC00740
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)                          CYC00750
C      PARAMETER (EPS=0)                                            CYC00760
C      DIMENSION X(NDIM),S(NDIM),Y(NDIM),Z(NDIM)                   CYC00770
C      DIMENSION Q(NDIM,0:KMAX-1),R(0:KMAX,0:KMAX)                 CYC00780
C      DIMENSION C(0:KMAX),GAMMA(0:KMAX),XI(0:KMAX-1)              CYC00790
C      EXTERNAL VECTOR                                              CYC00800
C      DO 40 IC=1,NCYCLE                                             CYC00810
C      IF (IPRES.EQ.1.OR.IPRES1.EQ.1) THEN                          CYC00820
C      WRITE(6,101) IC                                              CYC00830
101  FORMAT(/,' CYCLE NO. ',I3)                                     CYC00840
C      END IF                                                        CYC00850
C      NN=N                                                         CYC00860
C      IF (IC.EQ.1) NN=N0                                           CYC00870
C      IF (IPRES.EQ.1.OR.IPRES1.EQ.1) THEN                          CYC00880
C      WRITE(6,102) NN                                              CYC00890
102  FORMAT(/,' NO. OF ITERATIONS PRIOR TO EXTRAPOLATION IS',I3)   CYC00900
C      WRITE(6,103) KMAX                                             CYC00910
103  FORMAT(/,' WIDTH OF EXTRAPOLATION IS ',I3)                     CYC00920
C      END IF                                                        CYC00930
C      DO 20 J=0,NN-1                                               CYC00940
C      CALL VECTOR(X,Y,NDIM)                                         CYC00950
C      DO 10 I=1,NDIM                                                CYC00960
C      X(I)=Y(I)                                                     CYC00970
10  CONTINUE                                                         CYC00980
20  CONTINUE                                                         CYC00990
C      CALL MIERRE(METHOD,X,S,KMAX,KOUT,NDIM,Y,Z,VECTOR,Q,R,C,    CYC01000
C      *GAMMA,XI,RES,RES1,EPS,IPRES,IPRES1)                         CYC01010
C      IF (IC.EQ.1) RESP=R(0,0)                                     CYC01020
C      RESC=R(0,0)                                                  CYC01030
C      IF (RESC.LE.EPSC*RESP) RETURN                                CYC01040
C      DO 30 I=1,NDIM                                                CYC01050
C      X(I)=S(I)                                                     CYC01060
30  CONTINUE                                                         CYC01070
40  CONTINUE                                                         CYC01080
C      RETURN                                                        CYC01090
C      END                                                            CYC01100
C      SUBROUTINE MPPERRE(METHOD,X,S,KMAX,KOUT,NDIM,Y,Z,VECTOR,Q,R,C, CYC01120
C      *GAMMA,XI,RES,RES1,EPS,IPRES,IPRES1)                         CYC01130
C*****CYC01140
C      THIS SUBROUTINE APPLIES THE MINIMAL POLYNOMIAL EXTRAPOLATION (MPE) CYC01150
C      OR THE REDUCED RANK EXTRAPOLATION (RRE) METHODS TO A VECTOR CYC01160
C      SEQUENCE X0,X1,X2,..., THAT IS OFTEN GENERATED BY A FIXED POINT CYC01170
C      ITERATIVE TECHNIQUE.                                         CYC01180
C      BOTH MPE AND RRE ARE ACCELERATION OF CONVERGENCE (OR EXTRAPOLATION) CYC01190
C      METHODS FOR VECTOR SEQUENCES. EACH METHOD PRODUCES A TWO-DIMENSIONAL CYC01200
C      ARRAY S(N,K) OF APPROXIMATIONS TO THE LIMIT OR ANTILIMIT OF THE CYC01210
C      SEQUENCE IN QUESTION.                                         CYC01220
C      THE IMPLEMENTATIONS EMPLOYED IN THE PRESENT SUBROUTINE GENERATE CYC01230
C      THE SEQUENCES S(0,0)=X0,S(0,1),S(0,2),... .                 CYC01240
C*****CYC01250
C      AUTHOR : AVRAM SIDI                                           CYC01260
C      COMPUTER SCIENCE DEPARTMENT                                  CYC01270
C      TECHNION-ISRAEL INSTITUTE OF TECHNOLOGY                     CYC01280
C      HAIFA 32000,ISRAEL                                          CYC01290
C      E-MAIL ADDRESS: C55SIDI@TECHNION.BITNET                      CYC01300
C*****CYC01310
C      METHOD: IF METHOD.EQ.1, THEN MPE IS EMPLOYED. IF METHOD.EQ.2, THEN CYC01320

```

C		RRE IS EMPLOYED. INPUT.. (INTEGER)	CYC01330
C	X	: THE VECTOR X0. INPUT ARRAY OF DIMENSION NDM. (DOUBLE	CYC01340
C		PRECISION)	CYC01350
C	S	: THE APPROXIMATION S(0,K) PRODUCED BY THE SUBROUTINE FOR	CYC01360
C		EACH K. ON EXIT, S IS S(0,KOUT). OUTPUT ARRAY OF DIMENSION	CYC01370
C		NDIM. (DOUBLE PRECISION)	CYC01380
C	KMAX	: A NONNEGATIVE INTEGER.THE MAXIMUM WIDTH OF EXTRAPOLATION	CYC01390
C		ALLOWED. THUS THE NUMBER OF THE VECTORS X0,X1,X2,...,	CYC01400
C		EMPLOYED IN THE PROCESS IS KMAX+2 AT MOST. INPUT. (INTEGER)	CYC01410
C	KOUT	: A NONNEGATIVE INTEGER. KOUT IS DETERMINED BY A SUITABLE	CYC01420
C		STOPPING CRITERION, AND DOES NOT EXCEED KMAX.THE VECTORS	CYC01430
C		ACTUALLY EMPLOYED BY THE EXTRAPOLATION PROCESS ARE	CYC01440
C		X0,X1,X2,...,XP, WHERE P=KOUT+1. OUTPUT. (INTEGER)	CYC01450
C	NDIM	: DIMENSION OF THE VECTORS. INPUT. (INTEGER)	CYC01460
C	Y	: WORK ARRAY OF DIMENSION NDM. (DOUBLE PRECISION)	CYC01470
C	Z	: WORK ARRAY OF DIMENSION NDM. (DOUBLE PRECISION)	CYC01480
C	VECTOR:	A USER-SUPPLIED SUBROUTINE WHOSE CALLING SEQUENCE IS	CYC01490
C		CALL VECTOR(Y,Z,NDIM); Y,NDIM INPUT,Z OUTPUT.	CYC01500
C		Y,Z,NDIM ARE EXACTLY AS DESCRIBED ABOVE.FOR A FIXED POINT	CYC01510
C		ITERATIVE TECHNIQUE FOR SOLVING THE LINEAR OR NONLINEAR	CYC01520
C		SYSTEM T=F(T), DIM(T)=NDIM, Y AND Z ARE RELATED BY Z=F(Y).	CYC01530
C		THUS X1=F(X0), X2=F(X1), ETC.	CYC01540
C		VECTOR SHOULD BE DECLARED IN AN EXTERNAL STATEMENT IN THE	CYC01550
C		CALLING PROGRAM.	CYC01560
C	Q	: WORK ARRAY OF DIMENSION (NDIM,0:KMAX-1). FOR EACH K, ITS	CYC01570
C		ELEMENTS ARE THOSE OF THE ORTHOGONAL MATRIX OBTAINED FROM	CYC01580
C		QR FACTORIZATION OF THE MATRIX U	CYC01590
C		$U = (U_0   U_1   \dots   U_K)$ , $K=0,1,2,\dots$ ,	CYC01600
C		WHERE $U_0=X_1-X_0$ , $U_1=X_2-X_1$ , $U_2=X_3-X_2$ , ETC. OUTPUT. (DOUBLE	CYC01610
C		PRECISION)	CYC01620
C	R	: WORK ARRAY OF DIMENSION (0:KMAX,0:KMAX). FOR EACH K, ITS	CYC01630
C		ELEMENTS ARE THOSE OF THE UPPER TRIANGULAR MATRIX OBTAINED	CYC01640
C		FROM QR FACTORIZATION OF THE MATRIX U DESCRIBED ABOVE.	CYC01650
C		OUTPUT. (DOUBLE PRECISION)	CYC01660
C	C	: WORK ARRAY OF DIMENSION (0:KMAX). FOR EACH K, C FOR MPE IS	CYC01670
C		THE LEAST SQUARES SOLUTION OF THE SYSTEM $U^*C=0$ SUBJECT TO	CYC01680
C		THE CONSTRAINT $C(K)=1$ . (DOUBLE PRECISION)	CYC01690
C	GAMMA	: WORK ARRAY OF DIMENSION (0:KMAX). FOR EACH K, THE GAMMA'S	CYC01700
C		ARE SUCH THAT	CYC01710
C		$S(0,K)=GAMMA(0)*X_0+GAMMA(1)*X_1+\dots+GAMMA(K)*X_K$ .	CYC01720
C		FOR EACH K, GAMMA FOR RRE IS THE LEAST SQUARES SOLUTION OF	CYC01730
C		THE SYSTEM $U^*GAMMA=0$ SUBJECT TO THE CONSTRAINT	CYC01740
C		$GAMMA(0)+GAMMA(1)+\dots+GAMMA(K)=1$ . (DOUBLE PRECISION)	CYC01750
C	XI	: WORK ARRAY OF DIMENSION (0:KMAX-1). FOR EACH K, THE XI'S	CYC01760
C		ARE SUCH THAT	CYC01770
C		$S(0,K)=X_0+XI(0)*U_0+XI(1)*U_1+\dots+XI(J)*U_J$ , $J=K-1$ .	CYC01780
C		(DOUBLE PRECISION)	CYC01790
C	RES	: L2-NORM OF THE RESIDUAL FOR S(0,K) FOR A LINEAR SYSTEM	CYC01800
C		$T=A^*T+B$ (OR AN ESTIMATE FOR IT FOR A NONLINEAR SYSTEM	CYC01810
C		$T=F(T)$ ) FOR EACH K. ON EXIT, THIS K IS KOUT. OUTPUT.	CYC01820
C		(DOUBLE PRECISION)	CYC01830
C	RES1	: L2-NORM OF THE RESIDUAL ACTUALLY COMPUTED FROM S(0,K) FOR	CYC01840
C		EACH K. (THE RESIDUAL VECTOR FOR ANY VECTOR VEC IS TAKEN	CYC01850
C		AS $(F(VEC)-VEC)$ .) ON EXIT, THIS K IS KOUT. OUTPUT.	CYC01860
C		(DOUBLE PRECISION)	CYC01870
C	EPS	: AN UPPER BOUND ON RES/R(0,0), THE RELATIVE RESIDUAL FOR S,	CYC01880
C		USED IN THE STOPPING CRITERION. NOTE THAT $R(0,0)=L2-NORM$	CYC01890
C		OF THE RESIDUAL FOR X0, THE INITIAL VECTOR. IF, FOR SOME K,	CYC01900
C		$RES.LE.EPS*R(0,0)$ , THEN THE CORRESPONDING S(0,K) IS ACCEPTED	CYC01910
C		AS THE FINAL APPROXIMATION, AND THE SUBROUTINE IS EXITED	CYC01920
C		WITH KOUT=K. IF S(0,KMAX) IS NEEDED, THEN EPS SHOULD BE	CYC01930
C		SET EQUAL TO ZERO. INPUT. (DOUBLE PRECISION)	CYC01940
C	IPRES	: IF IPRES.EQ.1, THEN RES IS PRINTED FOR ALL K, $K=0,1,\dots$	CYC01950
C		OTHERWISE, IT IS NOT. INPUT. (INTEGER)	CYC01960
C	IPRES1	: IF IPRES1.EQ.1, THEN RES1 IS COMPUTED AND PRINTED FOR ALL	CYC01970
C		K, $K=0,1,\dots$ OTHERWISE, IT IS NOT. INPUT. (INTEGER)	CYC01980

```

C*****CYC01990
C THE ABOVE MENTIONED QR FACTORIZATION IS PERFORMED BY EMPLOYING CYC02000
C THE MODIFIED GRAM-SCHMIDT PROCESS. CYC02010
C*****CYC02020
      IMPLICIT DOUBLE PRECISION (A-H,O-Z) CYC02030
      PARAMETER (EPS1=1D-32,EPS2=1D-16) CYC02040
      DIMENSION X(NDIM),S(NDIM),Y(NDIM),Z(NDIM) CYC02050
      DIMENSION Q(NDIM,0:KMAX-1),R(0:KMAX,0:KMAX) CYC02060
      DIMENSION C(0:KMAX),GAMMA(0:KMAX),XI(0:KMAX-1) CYC02070
      IF (IPRES.EQ.1.AND.IPRES1.EQ.1) THEN CYC02080
        WRITE(6,301) CYC02090
301      FORMAT(/,' K RES RES1') CYC02100
        ELSE IF (IPRES.EQ.1.AND.IPRES1.NE.1) THEN CYC02110
        WRITE(6,302) CYC02120
302      FORMAT(/,' K RES') CYC02130
        ELSE IF (IPRES.NE.1.AND.IPRES1.EQ.1) THEN CYC02140
        WRITE(6,303) CYC02150
303      FORMAT(/,' K RES1') CYC02160
        END IF CYC02170
        DO 10 I=1,NDIM CYC02180
          Y(I)=X(I) CYC02190
10      CONTINUE CYC02200
        DO 250 K=0,KMAX CYC02210
          CYC02220
C      COMPUTATION OF THE VECTOR XJ, J=K+1, FROM XK, AND COMPUTATION OF UK CYC02230
C      CALL VECTOR(Y,Z,NDIM) CYC02240
C      DO 20 I=1,NDIM CYC02250
C      Y(I)=Z(I)-Y(I) CYC02260
20      CONTINUE CYC02270
C      CYC02280
C      DETERMINATION OF THE ORTHONORMAL VECTOR QK FROM UK BY THE MODIFIED CYC02290
C      GRAM-SCHMIDT PROCESS CYC02300
C      CYC02310
C      DO 50 J=0,K-1 CYC02320
C      SUM=0 CYC02330
C      DO 30 I=1,NDIM CYC02340
C      SUM=SUM+Q(I,J)*Y(I) CYC02350
30      CONTINUE CYC02360
C      R(J,K)=SUM CYC02370
C      DO 40 I=1,NDIM CYC02380
C      Y(I)=Y(I)-R(J,K)*Q(I,J) CYC02390
40      CONTINUE CYC02400
50      CONTINUE CYC02410
C      SUM=0 CYC02420
C      DO 60 I=1,NDIM CYC02430
C      SUM=SUM+Y(I)**2 CYC02440
60      CONTINUE CYC02450
C      R(K,K)=DSQRT(SUM) CYC02460
C      IF (R(K,K).GT.EPS1*R(0,0).AND.K.LT.KMAX) THEN CYC02470
C      HP=1D0/R(K,K) CYC02480
C      DO 70 I=1,NDIM CYC02490
C      Q(I,K)=HP*Y(I) CYC02500
70      CONTINUE CYC02510
C      ELSE IF (R(K,K).LE.EPS1*R(0,0)) THEN CYC02520
C      EEE=EPS1 CYC02530
C      WRITE(6,304) K,K,EEE CYC02540
304      FORMAT(/,' R(' ,I3,',',',I3,',') .LE.',1P,D8.1,'*R(0,0).',/) CYC02550
        END IF CYC02560
C      CYC02570
C      CYC02580
C      END OF COMPUTATION OF THE VECTOR QK CYC02590
C      CYC02600
C      IF (METHOD.EQ.1) THEN CYC02610
C      CYC02620
C      COMPUTATION OF THE GAMMA'S FOR MPE CYC02630
C      CYC02640

```

DO 90 I=K-1,0,-1	CYC02650
CI=-R(I,K)	CYC02660
DO 80 J=I+1,K-1	CYC02670
CI=CI-R(I,J)*C(J)	CYC02680
80 CONTINUE	CYC02690
C(I)=CI/R(I,I)	CYC02700
90 CONTINUE	CYC02710
C(K)=1D0	CYC02720
SUM=0	CYC02730
DO 100 I=0,K	CYC02740
SUM=SUM+C(I)	CYC02750
100 CONTINUE	CYC02760
IF (DABS(SUM).LE.EPS2) THEN	CYC02770
WRITE(6,311) K	CYC02780
311 FORMAT(/,' S( 0,',I3,' ) IS NOT DEFINED.',/)	CYC02790
GO TO 250	CYC02800
END IF	CYC02810
DO 110 I=0,K	CYC02820
GAMMA(I)=C(I)/SUM	CYC02830
110 CONTINUE	CYC02840
RES=R(K,K)*DABS(GAMMA(K))	CYC02850
C	CYC02860
C END OF COMPUTATION OF THE GAMMA'S FOR MPE	CYC02870
C	CYC02880
ELSE IF (METHOD.EQ.2) THEN	CYC02890
C	CYC02900
C COMPUTATION OF THE GAMMA'S FOR RRE	CYC02910
C	CYC02920
DO 130 I=0,K	CYC02930
CI=1D0	CYC02940
DO 120 J=0,I-1	CYC02950
CI=CI-R(J,I)*C(J)	CYC02960
120 CONTINUE	CYC02970
C(I)=CI/R(I,I)	CYC02980
130 CONTINUE	CYC02990
DO 150 I=K,0,-1	CYC03000
CI=C(I)	CYC03010
DO 140 J=I+1,K	CYC03020
CI=CI-R(I,J)*GAMMA(J)	CYC03030
140 CONTINUE	CYC03040
GAMMA(I)=CI/R(I,I)	CYC03050
150 CONTINUE	CYC03060
SUM=0	CYC03070
DO 160 I=0,K	CYC03080
SUM=SUM+GAMMA(I)	CYC03090
160 CONTINUE	CYC03100
DO 170 I=0,K	CYC03110
GAMMA(I)=GAMMA(I)/SUM	CYC03120
170 CONTINUE	CYC03130
RES=1D0/DSQRT(DABS(SUM))	CYC03140
C	CYC03150
C END OF COMPUTATION OF THE GAMMA'S FOR RRE	CYC03160
C	CYC03170
END IF	CYC03180
KOUT=K	CYC03190
IF (IPRES.EQ.1.AND.IPRES1.NE.1) THEN	CYC03200
WRITE(6,321) K,RES	CYC03210
321 FORMAT(I3,2X,1P,D15.2)	CYC03220
END IF	CYC03230
IF (RES.LE.EPS*R(0,0).OR.R(K,K).LE.EPS1*R(0,0)	CYC03240
* .OR.K.EQ.KMAX.OR.IPRES1.EQ.1) THEN	CYC03250
C	CYC03260
C COMPUTATION OF THE APPROXIMATION S(0,K)	CYC03270
C	CYC03280
XI(0)=1D0-GAMMA(0)	CYC03290
DO 180 J=1,K-1	CYC03300

```

      XI(J)=-XI(J-1)-GAMMA(J)
180 CONTINUE
      DO 190 I=1,NDIM
        S(I)=X(I)
190 CONTINUE
      DO 220 J=0,K-1
        HP=0
        DO 200 I=J,K-1
          HP=HP+R(J,I)*XI(I)
200 CONTINUE
        DO 210 I=1,NDIM
          S(I)=S(I)+HP*Q(I,J)
210 CONTINUE
220 CONTINUE
C
C   END OF COMPUTATION OF THE APPROXIMATION S(0,K)
C
      IF (IPRES1.EQ.1) THEN
C
C   EXACT COMPUTATION OF RESIDUAL L2-NORM.
C
      CALL VECTOR(S,Y,NDIM)
      RES1=0
      DO 230 I=1,NDIM
        RES1=RES1+(Y(I)-S(I))**2
230 CONTINUE
      RES1=DSQRT(RES1)
C
C   END OF EXACT COMPUTATION OF RESIDUAL L2-NORM.
C
      IF (IPRES.EQ.1) THEN
        WRITE(6,331) K,RES,RES1
331 FORMAT(I3,2X,1P,2D15.2)
      ELSE IF (IPRES.NE.1) THEN
        WRITE(6,332) K,RES1
332 FORMAT(I3,2X,1P,D15.2)
      END IF
      END IF
      IF (RES.LE.EPS*R(0,0).OR.R(K,K).LE.EPS1*R(0,0)) RETURN
      DO 240 I=1,NDIM
        Y(I)=Z(I)
240 CONTINUE
250 CONTINUE
      RETURN
      END

      SUBROUTINE VECTOR(X,Y,NDIM)
C*****
C THIS SUBROUTINE GENERATES THE VECTOR Z FROM THE VECTOR Y BY USING,
C E.G., A FIXED POINT ITERATION TECHNIQUE.
C*****
C IN THE PRESENT EXAMPLE THE ITERATIVE TECHNIQUE IS OF THE FORM
C  $Y=A_1*X+B_1$ . HERE  $A_1$  IS AN NDIM*NDIM SEPTADIAGONAL MATRIX SYMMETRIC
C WITH RESPECT TO BOTH OF ITS DIAGONALS, AND IS DEFINED AS
C  $A_1=(1-OMEGA)*I+OMEGA*A$ , WHERE OMEGA IS A SCALAR, I IS THE
C IDENTITY MATRIX, AND A IS THE MATRIX
C
C       | 5  2  1  1           |
C       | 2  6  3  1  1         |
C       | 1  3  6  3  1  1       |
C A = 0.06*| 1  1  3  6  3  1  1 | .
C       |     1  1  3  6  3  1  1 |
C       |           1  1  3  6  3  1  1 |
C       |               .  .  .  .  .  . |
C
C

```



C	B1 IS THE VECTOR DEFINED AS $B1 = \Omega * B$ , THE VECTOR B BEING CHOSEN	CYC03970
C	SUCH THAT THE SOLUTION OF THE SYSTEM $T = A * T + B$ IS THE VECTOR	CYC03980
C	$(1, 1, \dots, 1)$ .	CYC03990
C	THE ITERATIVE TECHNIQUE USED IS THUS RICHARDSON'S ITERATIVE	CYC04000
C	METHOD APPLIED TO THE SYSTEM $(I - A) * T = B$ .	CYC04010
C	*****	CYC04020
	IMPLICIT DOUBLE PRECISION (A-H,O-Z)	CYC04030
	PARAMETER (OMEGA=2D0,TAU=1D0-OMEGA)	CYC04040
	DIMENSION X(NDIM),Y(NDIM)	CYC04050
	N=NDIM	CYC04060
	$Y(1) = (5 * X(1) + 2 * X(2) + X(3) + X(4)) * 6D-2 + 46D-2$	CYC04070
	$Y(2) = (2 * X(1) + 6 * X(2) + 3 * X(3) + X(4) + X(5)) * 6D-2 + 22D-2$	CYC04080
	$Y(3) = (X(1) + 3 * X(2) + 6 * X(3) + 3 * X(4) + X(5) + X(6)) * 6D-2 + 1D-1$	CYC04090
	DO 10 I=4,N-3	CYC04100
	$Y(I) = (X(I-3) + X(I-2) + 3 * X(I-1) + 6 * X(I) + 3 * X(I+1) + X(I+2) + X(I+3)) * 6D-2$	CYC04110
	*       +4D-2	CYC04120
10	CONTINUE	CYC04130
	$Y(N-2) = (X(N) + 3 * X(N-1) + 6 * X(N-2) + 3 * X(N-3) + X(N-4) + X(N-5)) * 6D-2 + 1D-1$	CYC04140
	$Y(N-1) = (2 * X(N) + 6 * X(N-1) + 3 * X(N-2) + X(N-3) + X(N-4)) * 6D-2 + 22D-2$	CYC04150
	$Y(N) = (5 * X(N) + 2 * X(N-1) + X(N-2) + X(N-3)) * 6D-2 + 46D-2$	CYC04160
	DO 20 I=1,N	CYC04170
	$Y(I) = \text{TAU} * X(I) + \Omega * Y(I)$	CYC04180
20	CONTINUE	CYC04190
	RETURN	CYC04200
	END	CYC04210

CYCLE NO. 1

NO. OF ITERATIONS PRIOR TO EXTRAPOLATION IS 20

WIDTH OF EXTRAPOLATION IS 10

K	RES	RES1
0	4.75D-01	4.75D-01
1	5.36D-01	5.36D-01
2	1.52D-02	1.52D-02
3	1.93D-02	1.93D-02
4	4.23D-03	4.23D-03
5	3.79D-03	3.79D-03
6	1.41D-03	1.41D-03
7	1.00D-03	1.00D-03
8	5.16D-04	5.16D-04
9	3.04D-04	3.04D-04
10	2.00D-04	2.00D-04

CYCLE NO. 2

NO. OF ITERATIONS PRIOR TO EXTRAPOLATION IS 0

WIDTH OF EXTRAPOLATION IS 10

K	RES	RES1
0	2.00D-04	2.00D-04
1	9.57D-05	9.57D-05
2	9.59D-05	9.59D-05
3	4.58D-05	4.58D-05
4	4.42D-05	4.42D-05
5	1.68D-05	1.68D-05
6	1.91D-05	1.91D-05
7	6.49D-06	6.49D-06
8	7.22D-06	7.22D-06
9	2.56D-06	2.56D-06
10	2.90D-06	2.90D-06

CYCLE NO. 3

NO. OF ITERATIONS PRIOR TO EXTRAPOLATION IS 0

WIDTH OF EXTRAPOLATION IS 10

K	RES	RES1
0	2.90D-06	2.90D-06
1	1.18D-06	1.18D-06
2	1.38D-06	1.38D-06
3	6.20D-07	6.20D-07
4	6.64D-07	6.64D-07
5	2.43D-07	2.43D-07
6	2.63D-07	2.63D-07
7	8.58D-08	8.58D-08
8	9.15D-08	9.15D-08
9	3.95D-08	3.95D-08
10	4.17D-08	4.17D-08

CYCLE NO. 4

NO. OF ITERATIONS PRIOR TO EXTRAPOLATION IS 0

WIDTH OF EXTRAPOLATION IS 10

K	RES	RES1
0	4.17D-08	4.17D-08
1	2.44D-08	2.44D-08

2	2.40D-08	2.40D-08
3	1.29D-08	1.29D-08
4	1.24D-08	1.24D-08
5	5.49D-09	5.49D-09
6	5.39D-09	5.39D-09
7	1.95D-09	1.95D-09
8	1.96D-09	1.96D-09
9	8.71D-10	8.71D-10
10	9.27D-10	9.27D-10

CYCLE NO. 5

NO. OF ITERATIONS PRIOR TO EXTRAPOLATION IS 0

WIDTH OF EXTRAPOLATION IS 10

K	RES	RES1
0	9.27D-10	9.27D-10
1	5.14D-10	5.14D-10
2	5.46D-10	5.46D-10
3	2.74D-10	2.74D-10
4	2.71D-10	2.71D-10
5	1.17D-10	1.17D-10
6	1.09D-10	1.09D-10
7	4.64D-11	4.64D-11
8	4.28D-11	4.28D-11
9	2.07D-11	2.07D-11
10	2.19D-11	2.18D-11

CYCLE NO. 6

NO. OF ITERATIONS PRIOR TO EXTRAPOLATION IS 0

WIDTH OF EXTRAPOLATION IS 10

K	RES	RES1
0	2.18D-11	2.18D-11
1	1.25D-11	1.25D-11
2	1.26D-11	1.26D-11
3	7.23D-12	7.23D-12
4	6.32D-12	6.32D-12
5	3.29D-12	3.28D-12
6	2.85D-12	2.85D-12
7	1.27D-12	1.28D-12
8	1.15D-12	1.15D-12
9	5.79D-13	5.67D-13
10	5.67D-13	5.49D-13



## Report Documentation Page

1. Report No. <b>NASA TM-103240 ICOMP-90-20</b>		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle <b>Efficient Implementation of Minimal Polynomial and Reduced Rank Extrapolation Methods</b>				5. Report Date <b>August 1990</b>	
				6. Performing Organization Code	
7. Author(s) <b>Avram Sidi</b>				8. Performing Organization Report No. <b>E-5658</b>	
				10. Work Unit No. <b>505-62-21</b>	
9. Performing Organization Name and Address <b>National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191</b>				11. Contract or Grant No.	
				13. Type of Report and Period Covered <b>Technical Memorandum</b>	
12. Sponsoring Agency Name and Address <b>National Aeronautics and Space Administration Washington, D.C. 20546-0001</b>				14. Sponsoring Agency Code	
15. Supplementary Notes <b>Avram Sidi, Computer Science Department, Technion—Israel Institute of Technology, Haifa 32000, Israel and Institute for Computational Mechanics in Propulsion, Lewis Research Center (work funded by Space Act Agreement C-99066-G). Space Act Monitor: Louis A. Povinelli.</b>					
16. Abstract <b>The minimal polynomial extrapolation (MPE) and reduced rank extrapolation (RRE) are two very effective techniques that have been used in accelerating the convergence of vector sequences, such as those that are obtained from iterative solution of linear and nonlinear systems of equations. Their definitions involve some linear least squares problems, and this causes difficulties in their numerical implementation. In this work timewise efficient and numerically stable implementations for MPE and RRE are developed. A computer program written in FORTRAN 77 is also appended and applied to some model problems.</b>					
17. Key Words (Suggested by Author(s)) <b>Extrapolation methods Acceleration of convergence Vector sequences Iterative techniques</b>			18. Distribution Statement <b>Unclassified—Unlimited Subject Category 64</b>		
19. Security Classif. (of this report) <b>Unclassified</b>		20. Security Classif. (of this page) <b>Unclassified</b>		21. No. of pages <b>42</b>	
				22. Price* <b>A03</b>	

To be initiated by the responsible NASA Project Officer, Technical Monitor, or other appropriate NASA official for all presentations, reports, papers, and proceedings that contain scientific and technical information. Explanations are on the back of this form and are presented in greater detail in NHB 2200.2, "NASA Scientific and Technical Information Handbook".	<input checked="" type="checkbox"/> Original <input type="checkbox"/> Modified	(Facility Use Only) Control No. _____ Date _____
--	---	--

**I. DOCUMENT/PROJECT IDENTIFICATION** (Information contained on report documentation page should not be repeated except title, date and contract number)

Title: Efficient Implementation of Minimal Polynomial and Reduced Rank Extrapolational Methods

Author(s): Avram Sidi

Originating NASA Organization: Lewis Research Center

Performing Organization (if different): ICOMP

Contract/Grant/Interagency/Project Number(s): C99066G 505-62-21

Document Number(s): TM103240 Document Date: \_\_\_\_\_

(For presentations or externally published documents, enter appropriate information on the intended publication such as name, place, and date of conference, periodical or journal title, or book title and publisher: \_\_\_\_\_)

These documents must be routed to NASA Headquarters, International Affairs Division for approval. (See Section VIII)

**II. AVAILABILITY CATEGORY**

Check the appropriate category(ies):

Security Classification: ☐ Secret ☐ Secret RD ☐ Confidential ☐ Confidential RD ☒ Unclassified

Export Controlled Document - Documents marked in this block must be routed to NASA Headquarters, International Affairs Division for approval

☐ ITAR ☐ EAR

NASA Restricted Distribution Document

☐ FEDD ☐ Limited Distribution ☐ Special Conditions-See Section III

Document disclosing an invention

☐ Documents marked in this block must be withheld from release until six months have elapsed after submission of this form, unless a different release date is established by the appropriate counsel. (See Section IX)

Publicly Available Document

☒ Publicly available documents must be unclassified and may not be export-controlled or restricted distribution documents.

☐ Copyrighted ☐ Not copyrighted

**III. SPECIAL CONDITIONS**

Check one or more of the applicable boxes in each of (a) and (b) as the basis for special restricted distribution if the "Special Conditions" box under NASA Restricted Distribution Document in Section II is checked. Guidelines are provided on reverse side of form.

a. This document contains:

☐ Foreign government information ☐ Commercial product test or evaluation results ☐ Preliminary information ☐ Information subject to special contract provision

☐ Other - Specify \_\_\_\_\_

b. Check one of the following limitations as appropriate:

☐ U.S. Government agencies and U.S. Government agency contractors only ☐ NASA contractors and U.S. Government agencies only ☐ U.S. Government agencies only

☐ NASA personnel and NASA contractors only ☐ NASA personnel only ☐ Available only with approval of issuing office: \_\_\_\_\_

**IV. BLANKET RELEASE (OPTIONAL)**

All documents issued under the following contract/grant/project number \_\_\_\_\_ may be processed as checked in Sections II and III.

The blanket release authorization granted \_\_\_\_\_ is:

☐ Rescinded - Future documents must have individual availability authorizations.

☐ Modified - Limitations for all documents processed in the STI system under the blanket release should be changed to conform to blocks as checked in Section II.

**V. PROJECT OFFICER/TECHNICAL MONITOR**

Louis Povinelli 2600 Louis Povinelli / scl 7/27/90

Typed Name of Project Officer/Technical Monitor Office Code Signature Date

**VI. PROGRAM OFFICE REVIEW**

JOHN R. FACEY ☒ Approved ☐ Not Approved RP John R. Facey 8/16/90

Typed Name of Program Office Representative Program Office and Code Signature Date

**VII. INTERNATIONAL AFFAIRS DIVISION REVIEW**

☐ Open, domestic conference presentation approved. ☐ Export controlled limitation is not applicable

☐ Foreign publication/presentation approved. ☐ The following Export controlled limitation (ITAR/EAR) is assigned to this document: \_\_\_\_\_

☐ Export controlled limitation is approved.

International Affairs Div Representative \_\_\_\_\_ Title \_\_\_\_\_ Date \_\_\_\_\_

**VIII. EXPIRATION OF REVIEW TIME**

The document is being released in accordance with the availability category and limitation checked in Section II since no objection was received from the Program Office within 20 days of submission, as specified by NHB 2200.2, and approval by the International Affairs Division is not required.

Name & Title \_\_\_\_\_ Office Code \_\_\_\_\_ Date \_\_\_\_\_

Note: This release procedure cannot be used with documents designated as Export Controlled Documents, conference presentations or foreign publications

**IX. DOCUMENTS DISCLOSING AN INVENTION**

a. This document may be released on \_\_\_\_\_ Date \_\_\_\_\_ Installation Patent or Intellectual Property Counsel \_\_\_\_\_ Date \_\_\_\_\_

b. The document was processed on \_\_\_\_\_ Date \_\_\_\_\_ in accordance with Sections II and III as applicable NASA STI Facility \_\_\_\_\_ Date \_\_\_\_\_

**X. DISPOSITION**

Completed forms should be forwarded to the NASA Scientific and Technical Information Facility, P.O. Box 8757, B.W.I. Airport, Maryland 21240, with either (check box)

☐ Printed or reproducible copy of document enclosed

☐ Abstract or Report Documentation Page enclosed. The issuing or sponsoring NASA installation should provide a copy of the document, when complete, to the NASA Scientific and Technical Information Facility at the above listed address

IN-64 P 4/3  
JUL 31 7 35 AM '90  
RECEIVED

